碩 士 學 位 論 文

GPS와 가속도 센서 기반 정숙 운전 유도 시스템

2011年

漢 城 大 學 校 大 學 院

컴 퓨 터 工 學 科

컴 퓨 터 工 學 專 攻

金 善 浩

碩 士 學 位 論 文
指導敎授 李珉秭

GPS와 가속도 센서 기반 정숙 운전 유도 시스템

Driving Stability Management System
Using acceleration sensor and GPS

2010年 12月 日

漢城大學校 大學院
컴 퓨 터 工 學 科
컴퓨터工學專攻
金 善 浩

碩 士 學 位 論 文
指導敎授 李珉秭

GPS와 가속도 센서 기반 정숙 운전 유도 시스템

Driving Stability Management System
Using acceleration sensor and GPS

위 論文을 工學 碩士學位 論文으로 提出함

2010年 12月 日

漢城大學校 大學院
컴 퓨 터 工 學 科
컴퓨터工學專攻
金 善 浩

金善浩의 工學 碩士學位論文을 認准함

2010年 12月 日

審査委員長 _____ 印

審 査 委 員 _____印

審 査 委 員 _____印

목 차

【 표 목 차 】

【 그 림 목 차 】

제 1 장 서 론

1.1 연구 동기

교통사고는 운전자, 차량, 주변 교통상태, 도로 및 교통시설 등의 여러 요인이 복합적으로 작용하여 발생하지만 2009년의 경찰청 사고 통계자료 중 연간 교통사고 사망자 분석을 살펴보면 약 5,838건의 교통사고 중 안전 운전 불이행, 중앙선 침범, 신호 위반, 안전거리 미확보 등 운전자의 부주의에 의해 이루어진 사고가 약 5,402 건으로 약 90%이상을 차지하고 있음을 알 수 있다. 따라서 경찰청 사고 통계자료에서 보는 바와 같이 교통사고의 여러 요인 중 대부분의 사고자 운전자의 특성과 운전행태가 교통사고에 가장 큰 영향을 미치고 있음을 알 수 있다. 특히 이중 사업용 차량(시내, 시외, 기타버스)의 교통사고 사망자는 1,077 건으로 18.4%에 해당하는 것을 알 수 있다. 사업용 차량의 경우 다수의 승객의 안전을 책임져야 하므로 운전자의 위험운전은 더욱 심각한 실정이다.

운전자의 운전습관 개선 및 안전운전에 대한 사회적 요구, 사고 기록 등의 요구에 부응하기 위하여, 이미 다양한 형태의 디지털 주행기록계, 차량용 블랙박스 등이 국내외에서 출시되어 위험운전 감소방안으로 사용되고 있지만 디지털 주행기록계나 차량용 블랙박스 등은 사고에 대한 명확한 해석은 가능하지만 사고를 미연에 방지해 줄 수 있는 시스템은 현재 장착되어 있지 않다. 또한 현재 대부분의 차량에 속도 및 RPM을 이용하여 운전자에게 경고를 제공해주는 방식이기 때문에 현재 운전 습관에 대한 실시간 피드백이 이루어지지 않아 그 효율성이 매우 떨어지고 있는 실정이다.

본 본문에서는 기존의 차량용 블랙박스가 가진 모든 운행 정보 기록 기능을 기본으로 하고, GPS와 가속도 센서를 활용하여 차량 운행 정숙성을 수치화 하여, 승객의 승차감 개선을 유도하여 대중교통 수단의 편의성을 제고하고, GPS를 이용한 영상 및 광고를 출력하여 운행 사업자의 수익 개선

과 고객에 대한 정보 제공을 가능하고 무선 네트워크로 연결되어, 중앙에서 모든 차량의 운행 상황 및 내부 실시간 영상을 모니터할 수 있는 있는 시스템을 설계하였다. 그리그 GPS와 가속도 센서를 활용하여 차량 운행 정숙성을 수치화하는 정숙 운전 유도 시스템을 구현하였다.

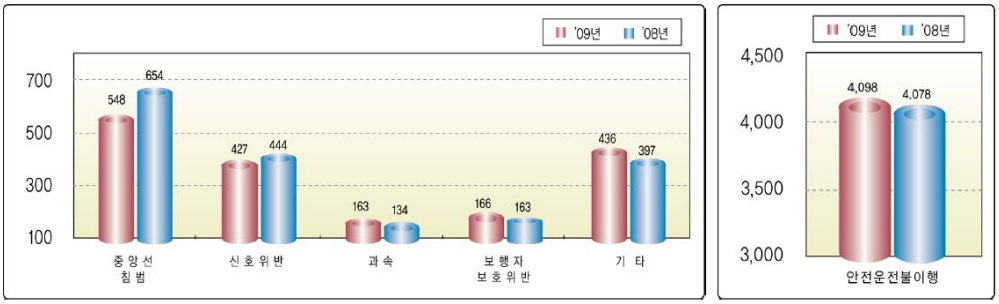


그림 1.1 교통사고 사망자 분석 (위반 유형별)

1.2 연구 목적

본 논문에서는 차량용 DVR 기능을 이용한 지능형 운전 안정성 관리 시스템에 대한 요구사항을 분석하고 요구 사항을 기반으로 소프트웨어 구조 설계와 그 중 GPS와 가속도 센서 기반 정숙 운전 유도 시스템을 구현하는데 목적이 있다. 본 논문의 세부적인 목적은 다음과 같다.

- **차량용 DVR 기능을 이용한 지능형 운전 안정성 관리 시스템 요구사항 분석**

  기존 연구되어온 다양한 운전 관련 시스템의 문제점과 차량용 DVR 기능을 이용한 지능형 운전 안정성 관리 시스템을 적용하기 위한 요구사항을 분석한다.

- **차량용 DVR기능을 이용한 운전 안정성 관리 시스템**

  플랫폼 독립적인 차량용 DVR 기능을 이용한 지능형 운전 안정성 관리 시스템 소프트웨어에 대하여 소개한다.

- **GPS와 가속도 센서 기반 정숙 운전 유도 시스템 구현**

  차량용 DVR 기능을 이용한 지능형 운전 안정성 관리 시스템 소프트웨어 구조를 기반으로 하여 PC base(윈도우)를 대상으로 소프트웨어 구조 중 하나인 GPS와 가속도 센서 기반 정숙 유도 시스템을 구현한다.

1.3 논문 구성

본 논문의 구성은 다음과 같다.

- **2장 차량용 DVR 기능을 이용한 지능형 운전 안정성 관리 시스템 연구 배경**

  차량용 DVR 기능을 이용한 지능형 운전 안정성 관리 시스템 관련 연구들에 대하여 기술한다.

- **3장 차량용 DVR기능을 이용한 운전 안정성 관리 시스템**

  차량용 DVR 기능을 이용한 지능형 운전 안정성 시스템에 대하여 기술한다.

- **4장 GPS와 가속도 센서 기반 정숙 운전 유도 시스템 구현**

  차량용 DVR 기능을 이용한 지능형 운전 안정성 시스템 중 GPS와 가속도 센서 기술을 활용한 정숙 운전 안정성 측정 기술과 중앙 모니터링 솔루션 일부를 실제 적용한 GPS와 가속도 센서 기반 정숙 운전 유도 시스템 구현에 대하여 기술한다.

- **5장 GPS와 가속도 센서 기반 정숙 운전 유도 시스템 파급효과 및 활용 방안결과 및 토의**

  본 논문에서 기술한 GPS와 가속도 센서 기반 정숙 운전 유도 시스템에 대하여 실제 상용화 방안에 대하여 기술한다.

- **6장 결론 및 향후연구**

  본 논문에서 설계 및 구현한 시스템에 대한 결론을 내리며 향후 연구 사항들에 대해 기술한다.

제 2 장 연구 배경

2.1 텔레매틱스

자동차 안에서도 운전자들은 교통정보뿐만 아니라 경제·문화 및 일반생활과 관련된 각종 정보 및 서비스를 제공받을 수 있게 되었고 지능형 교통 시스템의 개발로 인해 안전과 편의성이 크게 향상되게 되었다. 이러한 것을 가능하게 해주는 미래형 자동차 기술을 텔레매틱스라고 한다. 텔레매틱스(Telematics)는 무선통신을 이용해 차량과 정보센터를 연결, 차량 운행 중 요구하는 각종 정보와 서비스를 제공하는 기술이다. 텔레매틱스는 무선 음성, 데이터 통신과 인공위성을 이용한 위성위치정보 시스템(GPS)을 기반으로 정보를 주고받아, 위치측정 시스템과 무선통신망을 이용해 운전자와 탑승자에게 교통정보, 응급상황 대처, 원격차량진단, 인터넷 이용 등 각종 서비스를 제공할 수 있다. 여기에 최근의 텔레매틱스는 단순 응급 구난 중심으로 제공되던 서비스 개념에서 탈피, 무선인터넷의 개념을 도입한 이동통신 부가 가치 서비스로 정의된다.

하지만 텔레매틱스 시스템은 주로 전체적인 도로의 혼잡, 이상 상황, 교통 흐름 제어 등에 그 중심이 있으며, 그 정보를 교통 정보의 형태로 운전자에게 전달하는 것을 주요 목표로 하고 있다. 텔레매틱스는 또, 차량이 고장 나거나, 사고가 발생했을 때 무선 네트워크를 통해 자동으로 경찰, 보험회사, 정비회사 등에 보내는 기능을 포함하고 있으나, 본 논문처럼 운전자의 운전 습관에 관한 정보를 제공하지 않는다.

2.2 자동차용 블랙박스를 이용한 위험 인지

차량용 블랙박스는 정확한 사고 원인 규명을 위한 차량의 속도, 주행 방향, 차량 각 부의 작동 상태 등 차량의 운행에 대한 객관적/과학적 자료를 제공하며, 무선통신망을 활용하여 사고 차량의 위치 및 응급 상황을 자동 통보함으로써 원활한 사고 후속처리가 가능하다. 또한, 주행 및 사고 상황에 대한 정보 저장을 운전자가 인지하므로 위험운전 지양, 연료 절감, 교통사고 감소 효과 등의 기능을 가지고 있다. 즉, 위험 운전의 유형별 패턴 분석/인지를 통하여 실시간으로 위험을 경고하거나, 운전자의 성향 평가 및 차량 고장 진단 등의 정보를 제공하는 것이 가능하다.

블랙박스 저장 정보와 위험 운전 빈도, 교통사고 경험 유무 등의 정보들은 운전자의 위험 운전을 억제시키는 등 운전자와 운수회사 등에서 효율적인 활용이 가능하다(조준희, 이운성, 2007). 또한, 주행 정보의 수집/분석을 통하여 연료 절감, 배기가스 배출 절감을 위한 성능 분석을 할 수 있다. Sunwoo 등(Sunwoo et al., 2003)은 서울 도심의 주행 노선을 선정하고 블랙박스를 장착한 실차 실험을 통하여 다양한 연령층의 남녀 운전자의 주행 데이터를 수집하였다. 그리고 수집된 주행 데이터의 분석과정을 통해 연료 소비율에 영향을 미치는 주행 변수와 운전자의 성별, 연령별, 주행 노선별 연비 절감을 위한 기준을 제시하였다.

또한 Mundke 등(Mundke et al., 2006)도 상용 버스에 블랙박스를 장착한 실차 실험과 시뮬레이션 프로그램(Racer)를 이용하여 다양한 주행 데이터를 수집하였다. 그리고 주행 특성과 연료 소비율의 관계에 대하여 데이터마이닝 등의 통계적 기법을 활용하여 연료절감 운

전 및 운전자 평가에 대한 기준을 제시하였다. Alessandrini 등(Alessandrini et al., 2006)은 GPS를 이용한 차량의 위치 및 거동 정보와 OBD-II 커넥터를 통한 연료 소비, 엔진 출력, 공연비 등의 주행 정보를 실차 실험을 통하여 수집하고 운전 유형과 도로형상에 따른 차량 거동에 대한 분석을 통하여 차량 성능, 에너지 소비와 배기가스 배출 절감 방안 등에 대하여 연구한 바 있다. 신용균 등(신용균 등, 2006)은 과속 운전과 주행 습관 등에 대한 심리적 요인을 분석하여 운전자 재교육 방안에 대하여 제시한바 있다.

하지만 블랙박스는 차량의 사고 데이터뿐만 아니라 주행 기록계의 진화된 형태로서 차량의 주행 거동에 대한 각종 데이터의 수집·저장이 가능하므로 수집·저장된 정보의 분석 과정을 통하여 위험 운전을 판별할 수 있으나, 실시간으로 운전 인지를 판단하는 기능은 제공하지 않고 있다.

2.3 위험 운전 유형 분류 및 임계값 선정

차량용 블랙박스는 평상시와 사고시의 정보를 저장하는 장치로써 그 정보는 다방면에서 활용되고 있으면, 성능개선을 위한 연구가 활발히 진행되고 있다(한인환 등, 2007 a). 위험 운전 유형을 급제동, 급가속, 급선회, 급차선 변경의 네 가지로 분류하고 차량용 블랙박스를 장착한 실차 실험을 통해 주행 데이터를 수집·분석하여 위험운전을 인지하는 알고리즘을 제시한다. 유형에 따른 알고리즘은 차속에 따른 센서 측정치의 변화량과 도달 시간을 기준으로 구성하였다. 급제동과 급가속 유형에서는 종 가속도가 짧은 도달 시간 동안 급격히 감소하거나 증가하는 경향이 나타났다. 급선회와 급차선 변경은 급선회의 경우에는 긴 주기, 급차선 변경은 짧은 주기 동안 선회속도의 변화가 크게 나타났다. 도출된 기록 데이터는 위험운전 상황 발생순산 센서 측정값을 처리하는 구조가 간단하고 데이터 항목이 적은 온라인과 일정 구간 또는 장시간 주행 데이터를 차량 운행 종료 후 저장부에서 추출하는 오프라인으로 구분할 수 있다. 이 데이터의 분선은 운전자 및 차량 관리 시스템을 구축하는데 활용할 수 있다(한인환 등, 2007 b).

하지만 이러한 실험을 통한 세부적인 위험운전 유형과 임계값을 제시하였으나, 본 논문에서처럼 운전자의 운전습관에 대하여 점수화하지 않는다. 따라서 운전자에게 실시간 피드백이 이루어지지 않아서 운전자와의 현실적인 교감이 부족하다.

2.4 차량용 블랙박스를 이용한 위험운전 인지

기존 연구, 경찰청 자료 및 보험사의 사고 유형 분류 자료를 이용하여 위험운전 유형을 분류하고 분류된 위험운전 유형이 현실성 있는 위험운전 유형인지 검증하기 위하여 Data-logger을 개발하였다. Data-logger는 시험차량, Gyro, GPS 및 DAQ 시스템, Rack 시스템, 영상 데이터 취득 시스템, 전원 공급 장치와 운전자 인지 검출 시스템으로 이루어진 하드웨어와 Real-time 모듈, Replay 모듈, Export 모듈로 이루어진 소프트웨어로 구성되어 개발되어 있으며, 교통안전 및 인간 공학적인 연구 활동을 위한 첨단 종합 도구로서 실용성이 매우 높다고 할 수 있다. 향후 추진하고자 하는 운전 형태 분석 및 위험운전 판단 알고리즘 개발에 효과적인 도구로 활용할 수 있을 것으로 판단된다.

또한 교통사고의 감소와 안전운전 도모를 위한 디지털 주행기록계나 차량용 블랙박스 같은 다양한 시스템이 개발되고 있다. 차량의 동적 거동과 차량의 상태 및 운전자의 조작정보를 모니터링하고 기록하여 해석하는 장치인 차량용 블랙박스(EDR: event data recorder)는 자동차 충돌 사고 전후의 상황을 기록하여 신속한 사고 후처리 및 과학적인 사고 해석을 위한 자동차 전장품으로 현재 도입 단계 내지는 시행 초기에 잇는 기술이다.

한발 더 나아가 유럽에서는 운전자의 운전형태와 차량의 거동 및 주변 환경을 실시간으로 분석하여 운전자에게 다양한 정보를 제공해 주는 첨단 운전자 지원시스템의 연구가 진행되고 있다. 특히 차선과 차량, 교통신호와 같은 주변 환경을 인식하기 위하여 Vision 센서, 초음파 센서, 레이더 센서, 레이저 센서 등과 같은 다수의 다양한 센서를 차량에 접목시키고 있다.

Vetronix社는 CDR(crach data retrieval)이라는 자동차 충돌사고 처리 시스

템을 개발하여 차량의 DLC(diagnostic link connector) 혹은 에어백 모듈로부터 CDR 소프츠웨어를 이용하여 충돌 사고 데이터를 직접 다운로드 할 수 있다.

또한, 주행 중 차선과 주변차량과 같은 주변상황을 인식하기 위한 연구는 첨단운전자원시스템 개발을 목표로 다양한 자동차관련 업계 및 연구소 등에서 활발한 연구가 진행 중이다. 일본의 Denso社는 ADAS(Advanced Driver Assistance System)을 구현하기 위하여 Sensing System을 제안하고 있다.

국내에서 또한 첨단 시스템의 하나로 선행 차량을 인식하여 차량의 주행 속도 등을 조절 할 수 있는 스마트 크루즈 컨트롤(SCC) 시스템이 개발되어 차량에 적용되어 출시되고 있다.

스마트 크루즈 컨트롤(SCC) 시스템은 선행 차량을 인식함으로서 내 차량의 주행속도를 조절하고, 방향 지시등 없이 차선을 이탈하는 경우 운전자에게 경고를 주어 사고를 미연에 방지한다.

하지만 이러한 실험을 통한 위험운전 유형 검증과 위험을 인지하여 실시간 주행 속도를 감소해 주는 등 위험을 확인 또는 사고로부터 운전자를 보호해 주는 시스템은 제시하였으나 본 논문에서처럼 운전자 스스로 운전습관을 체크할 수 없다. 따라서 실시간 피드백이 이루어지지 않아서 운전자와의 현실적인 교감이 부족하다.

2.5 자동차용 블랙박스 업체의 기술/제품 동향

차량용 다기능 DVR과 중앙 모니터링 솔류션, 광고 및 영상 저작 및 송출 솔루션을 포함하는, 다양한 기술의 복합체로 주로 블랙박스 기능을 제공하는 국내·외 업체들의 제품들의 동향을 분석하였다.

2.5.1 국내 기술 현황

국내에서는 사고 해석 및 후속 처리에 대한 관심이 높아지면서, 디지털 운행기록계 및 차량용 블랙박스에 대한 연구개발이 진행되어 관련 시제품들이 출시되고 있다. 현재 HK이카, 카스포, 모비콘 등이 제품 기술을 보유하고 있다.

가. 모비콘

모비콘의 운전자정보시스템은 상용자동차의 전자엔진 제어장치로 부터 각종 파라미터 데이터를 실시간으로 수집하여 데이터를 저장하거나 모니터링 할 수 있어 각종 장치들의 제어나 동작상태를 확인할 수 있는 시스템으로 고장진단 정보, 실시간 주행정보를 제공한다.

나. 피엘케이테크놀러지

차선이탈 경고, 도로정체 및 신호 대기시 앞차 주행시 경고하는 스톱&고 기능과 영상 블랙박스 기능 등을 갖춘 장치를 제조한다.

다. HK이카

현대-기아차 사내벤처로 출발한 HK이카의 차량용 블랙박스는 차량운행기록을 디지털 방식으로 저장하며, 사고 전·후 일정시간의 조향각도, 차량 속도, 엔진 회전수, 차량 전장비의 구동 여부 등을 저장하여 사고 당시의 차량 상황을 저장하고, 평시에는 각 장비의 정상 작동 여부를 점검하는 기능도 겸비하고 있다. 주요 기능은 아래와 같다.

· 사고 자동 인식 기능

전후/좌우 모든 방향에서 발생하는 충돌 사고를 감지 및 충돌 사고와 충돌 유사 사항을 정확히 구별.

· 차량 운행 데이터의 기록

차량 운행시간, 운행거리, 최고속도, 평균속도, 급가속, 급제동, 운행 횟수, 시간대별 과속 상황, 공회전, 전조등과 미등의 조작 여부, 위험 운전 여부 등 잘못된 운전자의 운전 습관 교정.

· 기록된 운행 데이터의 분석

충돌량, 충돌 각도, 충돌 속도 등 주요 충돌 데이터를 분석하며, 종횡가속도, 방향 각속도, 방향각, 회전 반경, 차량 궤적, 각도 등 사고 전, 후의 차량 거동과 운전자 조작 상황을 각종 그래프 및 동영상으로 재현하고, 사고 원인을 분석.

· 자동 사고통보

자동 사고통보 기능을 가지는 모듈형 블랙박스는 무선 통신망(CDMA, GSM)을 통하여 사고 발생 시 사고 발생지점과 충돌 정보를 관제센터에 자동으로 통보하여, 인원 및 차량의 손상 정도에 따라 긴급 구난 서비스 및 교통 정보 제공 서비스 시스템을 가동.

라. 카스포

카스포의 제품은 디지털식 기록매체를 사용하여 순간 최고속도를 1초 단위로 저장하여 순간 최고속도 운행거리 등의 운행정보를 실시간으로 저장하며, 운전습관에 해당되는 과속, 엔진 과회전, 장시간과속, 급가속, 급제동 등의 운전 자료를 DB화하여 효율적인 운행관리 및 운전원 관리를 할 수 있으며, 또한 차량 운행정보를 기록하는 디지털 메모리 팩은 기록지를 대체하여 기존의 기계식 또는 전기식 타코 그라프에 비해 훨씬 편리하며 사고발생시 속도 정밀 분석이 가능하여 차량용 블랙박스로의 활용이 가능하다.

마. 유비원

유비원 제품은 카메라를 통해 운전자 및 탑승자의 표정과 외부 상황 등 사고당시의 영상을 고화질로 촬영하고, 내장형 GPS를 장착하여 시간, 위치, 속도를 저장한다. 센서로 주행 중 차량의 충격, 급제동, 급발진, 급커브 같은 차량의 전후, 좌우, 상하 진동상황 등을 분석한다. 사고전후로 최대 30초의 영상과 오디오를 저장하고, 운전자가 Emergency 버튼을 눌러 임의적으로 영상을 촬영할 수 있다.

시동을 걸면 작동되며 운행 중엔 계속적으로 다른 장면이 기록되지만, 사고가 나는 순간 충격이 가해지면 차량내부, 외부의 전후, 좌우의 상황 과 GPS를 통해 차량속도, 시간, 위치 등이 상세하게 기록된다. 마이크가 내장되어 있어 운전자의 대화나 충격에 의한 소리 등도 저장한다. 이렇게 수집된 운전정보로 사고영상 재생은 물론 GPS정보와 차량 가속정보를 확인한다.

2.5.2 국외 기술 현황

국외의 자동차용 블랙박스 제품은 대만의 로얄텍, 일본의 자동차 연구소, 유럽의 SIEMENS VDO, 미국의 Vetronix, 로드세이프티, 드라이브캠 등의 업체와 GM, 포드, 도요다, 혼다, 사브 등의 자동차 업체에 의해서 개발되었다.

가. 로드세이프티

로드세이프티는 10대 운전자를 둔 부모들을 상대로 판매하고 있으며 부모들은 자녀의 운전성향을 확인하고 교육할 수 있으며, 과속이나 급정지 시, 안전벨트 미착용 시 경고음을 통해 운전자에게 위험을 알린다. 또한, 안전운전시스템을 통해 과속, 급제동, 급조향 등의 운전정보를 저장하고 DB화하여 차량 관리에 용이하며, 사고 상황에서 저장된 정보를 활용하여 사고 해석이 가능하다.

나. 드라이브캠

드라이브 캠의 드라이빙 피드백 시스템(Driving Feedback System)을 통해 위험한 운전습관을 식별하며, 자동차 백미러에 장착되어 사고 시 차량 내부 영상 및 음성 데이터를 저장한다. 저장 데이터는 컴퓨터로 자동 전송되며, 사고 상황을 분석할 수 있다. 또한 사고 정보를 보험사 등으로 직접 통보할 수 있다.

다. SIEMENS VDO

VDO는 SAAB, BENZ 등 유럽의 각 자동차 회사에 공급하여 양산 적용 중에 있으며 EU 차원에서 표준화/법제화 검토 중에 있다.

라. Vetronix

Vetronix는 차량용 블랙박스 하드웨어장치, 충돌감지 소프트웨어, 자동차 충돌 사고처리시스템을 개발했다. 차 내에 장착되는 차량시스템을 제공하는 것 이외에 Vetronix는 위험한 충돌 정보와 의학적인 데이터를 무선 및 인터넷을 통해 긴급통신센터로 보내는 텔레마틱스 기본 시설을 구축했다. 이러한 고유 능력을 바탕으로Vetronix는 자동충돌감지, GPS 위치확인, 충돌정도 보고, 효율적이며 효과적인 도로 상에서의 의학치료를 가능케 하는 의학정보 제공 등을 포함하는 포괄적인 턴키방식의 솔루션을 제공한다.

마. 로얄텍사

로얄텍사의 블루 GPS는 국내에서도 GPS 교통정보 제공 네비게이션으로 판매되고 있으며, GPS 기반의 위치 정보 제공의 역할이 주요 기능으로 다양한 모바일 데이터를 제공하며, 사용자의 기호에 따라 차량 네비게이션, 보행 정보 제공 등으로 활용 가능하다. 차량 이동 경로, 속도, 시간, GPS 위성 수신 상태 등의 모바일 데이터를 PC, 노트북, PDA 등에서 다시 재현해 볼 수 있어 차량 장착을 통해 네비게이션 활용 시 사고 해석 정보를 제공한다.

제 3 장 차량용 DVR기능을 이용한 운전 안정성 관리 시스템

3.1 시스템 개요

차량의 보안을 위한 DVR이 CCTV 녹화 시 차량 안에 장착되어 GPS기능과 급가속, 급회전, 급감속의 가속도 신호를 함께 녹화하고 이 신호를 이용한 승차감 분석 및 운전의 안정성과 승차지수를 측정하여 정량적으로 운전자의 운전습관을 개선할 수 있는 프로그램을 함께 제공한다.

그림 3.1 시스템 개요

또한 차량 DVR의 출력 화면에 비디오, 실시간 TV, 광고의 정보 채널을 보여주면서 차량이 지나가는 지역의 특화된 광고/이벤트의 실시간 영상을 보여주도록 한다. 이때 광고, SMS 타이틀 및 비디오 클립을 스케줄 방송을 할 수 있도록 하는 자막송출기 및 비디오 송출 서버를 내장하는 기술로 방송과 보안 기술을 접목한 시스템이

된다. 본 시스템은 Wibro/Wimax/HSDPA 등 무선 네트워크를 통해 광고 내용이나 비디오 콘텐츠를 실시간으로 Update하게 됨으로써 움직이는 방송실이 될 수 있다.

따라서 본 시스템은 운행정보, 여러 카메라 영상을 저장하는 운행기록, 보안 영상 DVR 기능, 이를 중앙에서 모니터링하고 상황을 볼 수 있는 네트워크 DVR, 그리고 승차감, 난폭운전지수를 만들어내는 센서 장치와 알고리즘, CCTV 장치와 방송 송출 서버 기능을 통합하는 장치로 무선 네트워크 환경의 WiBro, HSDPA의 네트워크를 통한 방송 보안 시스템이 운영되는 최첨단 보안, 방송, 승차감 분석의 통합 시스템이다.

그림 3.2 최종 상품의 이미지

3.2 시스템 구성

차량용 DVR기능을 이용한 운전 안정성 관리 시스템은 차량용 다기능 디지털 비디오 레코더 (DVR) 및 그 운영방법 기술을 활용한 지능형 운전의 안정성 관리 시스템이다. 차량용 다기능 DVR 장치는 복수의 감시카메라에서 촬영된 영상신호, 무선 네트워크로 수신된 영상 신호를 인코딩/디코딩하는 영상처리수단과, 차량의 위치정보를 수신하여 위치별 광고 및 부가정보를 출력하는 광고처리수단, 차량의 모든 운행 정보를 처리하는 수단을 포함한다. 이 디지털 비디오 레코더에는 운행 정보, 광고 및 부가정보와 영상신호를 저장하기 위한 하드 디스크가 포함된다. 차량용 DVR의 하드 디스크는 영상신호와 광고 및 부가정보 외에도 미리 입력된 각종 멀티미디어 데이터를 저장한다. 또한 광고 및 모든 영상 및 데이터는 스케줄에 의해 또는 중앙의 제어신호에 따라 출력장치에 출력되고, 차량이 소정위치에 근접하면 광고 및 부가정보가 모니터를 통해 출력된다.

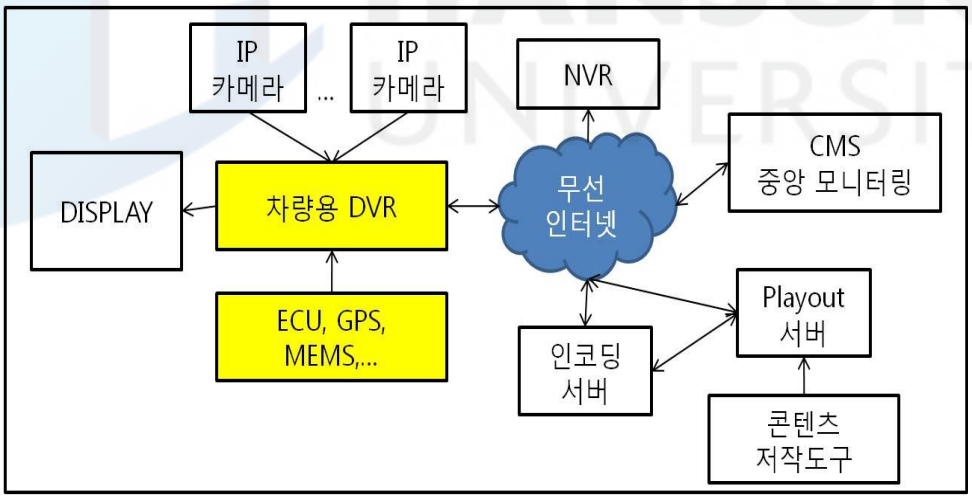


그림 3.3 전체적인 시스템의 구성

즉, 차량 DVR 시스템은 기본적으로 대형 차량을 고려하여 여러 대의 카메라, MEMS/네비게이션 센서 처리 모듈, 비디오 Playback 모듈 등으로 구성된다. 이들의 신호는 고화질 정보는 자체 시스템에도 기록되지만, 유선 무선 네트워크를 통해 중앙 제어실에 정보가 실시간으로 보내져 모니터링이 되어야 하기 때문에 중앙 모니터링을 위한 솔루션이 필요하다.

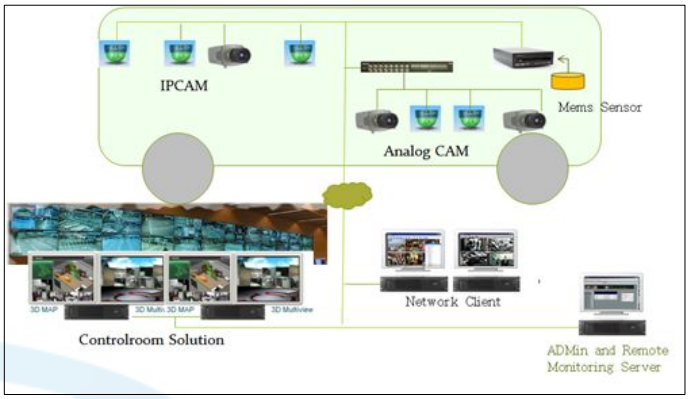


그림 3.4 차량용 DVR 시스템 구성도

또한 차량용 DVR의 광고 및 실시간 영상 출력을 지원하기 위해서는 차량 내에서의 실시간 영상을 제작 송출하기 위한 타이틀 제작 및 Authoring 시스템과 인코딩 서버, Playout 서버가 필요하다.

3.2.1 중앙 모니터링 솔루션 (Central Monitoring Solution)

수많은 차량용 DVR을 모니터하기 위해서는 중앙 모니터링 솔루션이 필요하며, 이동 중인 차량이 모니터링 대상이므로 기존 CMS에 GPS 연동이 가능하도록 개발한다.

그림 3.5 중앙 모니터링 솔루션 구성도 및 예상 결과 화면

3.2.2 GPS와 가속도 센서 기술을 활용한 정숙 운전 안정성 측정 기술

고속주행 등 가속도 범위내의 급정거, 급가속, 급회전을 정확히 측정하고 이들 상황을 분석하기 위하여 GPS 및 가속도 센서 기술을 활용한다. 차량용 DVR에서는, 이들 신호와 신호의 처리 데이터를 통해 정숙 운전 안정성(정숙도)을 측정하고 평가하는 모듈을 구현하고 이를 차량용 DVR에 인테그레이션 한다.

차량운행 영상기록 장치는 차량 주행 또는 정차 중 사고 발생 시 사고 당시의 영상과 GPS에 의한 사고 차량의 위치정보 등을 내부 저장매체에 자동 저장한 후 선명한 화질로 재생해 정확한 사고 원인 규명에 도움을 줄 수 있어야 한다.

또한 차량 사고와 같은 충격이 발생하는 상황 이외에 일반적으로 주행하고 있는 모든 영상을 저장해 운전자의 운행습관을 개선하는 참고 자료로 활용 가능하다. 사고 발생 시 영상 뿐 아니라 GPS 정보(위치, 속도, 시간)를 통해 보다 정확한 참고자료로 활용 가능하고, 녹화한 영상들을 시간, 속도, 이벤트별로 검색할 수 있으며, 검색된 영상의 궤적을 지도에 표시해 이동 상황을 한눈에 알아볼 수 있도록 개발한다.

그리고 운전자의 동선 추적 기능, 센서 정보로 운전 중 발생사고 정보 감지, 차량 운행 속도 파악, 운전자의 운전습관 및 방법 분석 등을 개발한다.

3.2.3 다채널 인코딩 Server

여러 운행 사업자가 필요로 하는 다양한 광고와 실시간 영상을 지원하기 위해서는 다채널 실시간 엔코딩을 지원하여야 한다. 또한, 보안 영상에서 인물 식별등을 자동화할 수 있는 수준의 고화질 영상을 제공할 수 있는 HD급에 대한 연구를 진행되어야 한다. 그리고 낮은 대역폭의 네트워크 환경에서도 다채널 아날로그 카메라 영상을 처리하여 고품질의 영상 스트리밍을 제공하여야 한다.

3.2.4 타이틀 제작 및 Authoring 시스템

차량용 DVR에서 출려될 영상을 제작하기 위한 저작 시스템은 타이틀 제작, 비디오 편집, SMS 메시징 전달 등의 방송용 서버 기능을 단순화시켜 제공한다. 이와 같은 자동송출과 타이틀 메시징이 GPS와 연동되어 지역의 정보에 따라 움직이도록 하는 광고 삽입기능과 Playout 서버 기능을 통합하는 시스템 운영이 가능하다.

그림 3.6 타이틀 제작 및 Authoring 시스템

또한 타이틀 제작, 비디오 편집, SMS 메시징 전달 등의 방송용 서버 기능을 차량용 DVR을 위한 방송에 맞도록 customize 하여 구현한다.

제 4 장 GPS와 가속도 센서 기반 정숙 운전 유도 시스템 구현

4.1 시스템 개발 환경

4장에서는 3장에서 소개한 차량용 DVR기능을 이용한 운전 안정성 관리 시스템 중 GPS와 가속도 센서 기술을 활용한 정숙 운전 안정성 측정 기술과 중앙 모니터링 솔루션 일부를 실제 적용한 내용을 다룬다. 본 논문에서 구현한 시스템 환경은 아래와 같다.

		차량 DVR 측 개발
하드웨어		인텔 코어 2 듀오
운영체제		윈도우즈
개발툴	에디터	Visual C++ 6.0
	버전관리	서브버전
	버그 관리	엑셀
	테스팅	소스 인스펙션
개발 언어		C

		CMS 모니터링 시스템 측 개발
하드웨어		인텔 코어 2 듀오
운영체제		윈도우즈
개발툴	에디터	Qt
	버전관리	서브버전
	버그 관리	엑셀
	테스팅	소스 인스펙션
개발 언어		C++

4.2 GPS와 가속도 센서 기반 정숙 운전 유도 시스템 구조

차량용 DVR은 가속 센서와 GPS 정보를 이용하여 급정거, 급가속, 급회전을 정확히 측정하고 이들 상황을 분석할 수 있는 난폭운전 측정 시스템 등 운전 안정성을 저장하고, 클라이언트와 통신을 하는 소프트웨어를 독립적인 형태로 개발하였다. 그리고 CMS에서는 차량용 DVR과 통신하여 위치 및 운행 정보를 실시간으로 확인할 수 있도록 하였으며, 저장된 운행 정보를 이용하여 차량의 운행 기록, 궤적도 확인할 수 있도록 모니터링 모듈을 개발하였다.

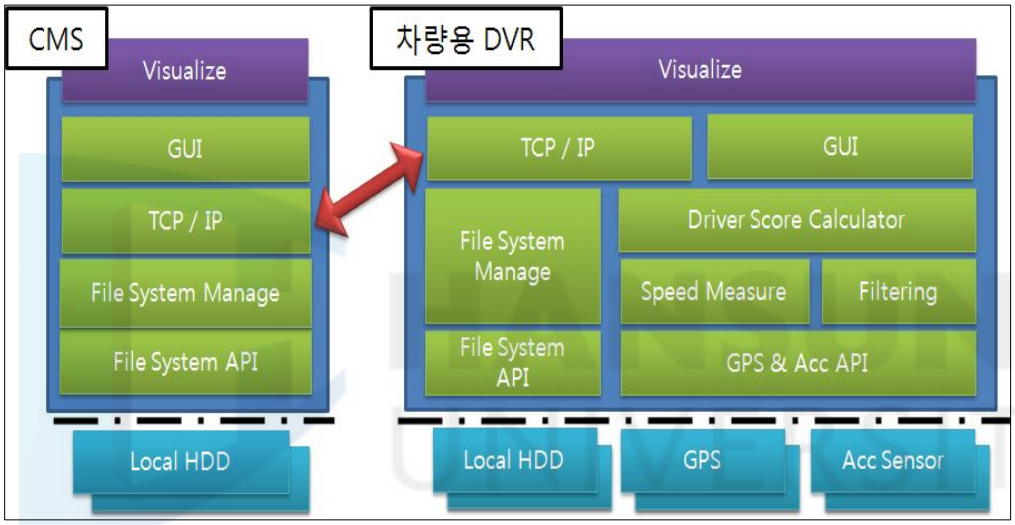


그림 4.1 CMS 및 차량용 DVR 소프트웨어 구조

4.2.1 차량용 DVR 시스템

ECU로부터의 정보와 GPS 체계에서 얻을 수 있는 정보를 이용하여 차량의 모든 운행 상황을 지속적으로 기록한다. 이 과정은 통상적인 차량 운행 기록기와 거의 유사하다. 본 논문의 경쟁력은 위 정보와 함께 가속도 기반 관성 센서 체계의 정보를 이용하여 정숙 운전 정숙도를 평가하여 다른 운행 정보와 함께 기록하고, 이를 수치 점수화 하여 화면에도 표시함으로써 승객들도 볼 수 있도록 한다. 이 과정을 통해, 운행 기록에 대한 저장뿐만 아니라, 운전 정숙도를 개선하여 특히 대중교통 수단 이용의 편의성을 높인다.

4.2.2 CMS 중앙 모니터링 시스템

자사의 운행 상황을 모니터하기 위해서는 수많은 차량용 DVR을 동시에 감시할 수 있는 중앙 모니터링 시스템이 필요하다. 이 시스템은 기존의 CMS 시스템에 추가될 부분으로 GPS를 이용한 차량 위치, 현재 운행 기록 상황을 Map에 표시할 수 있도록 구현했다.

4.3 차량용 DVR 구현

4.3.1 차량용 DVR 활동 다이어그램

GPS와 가속도 센서를 이용하여 운전 정숙도를 평가하여 다른 운행 정보와 함께 기록하고, 이를 수치 점수화하여 CMS 측에 전달하는 과정과 화면에 표시하는 과정을 그림 4.2로 보여준다.

수행 과정을 보면 프로그램이 시작되면 GPS 모듈과 Sensor 모듈을 통하여 주기적으로 GPS 데이터, 속도 및 Sensor 데이터를 읽어온 후 Calculate Drive Score 모듈을 통하여 운전 점수를 수치화하고 운행 정보와 함께 로컬에 txt 파일로 저장하고 승객들에게 보여준다. 마지막으로 CMS 모니터링 시스템에게 실시간 또는 이전 정보를 전달한다.

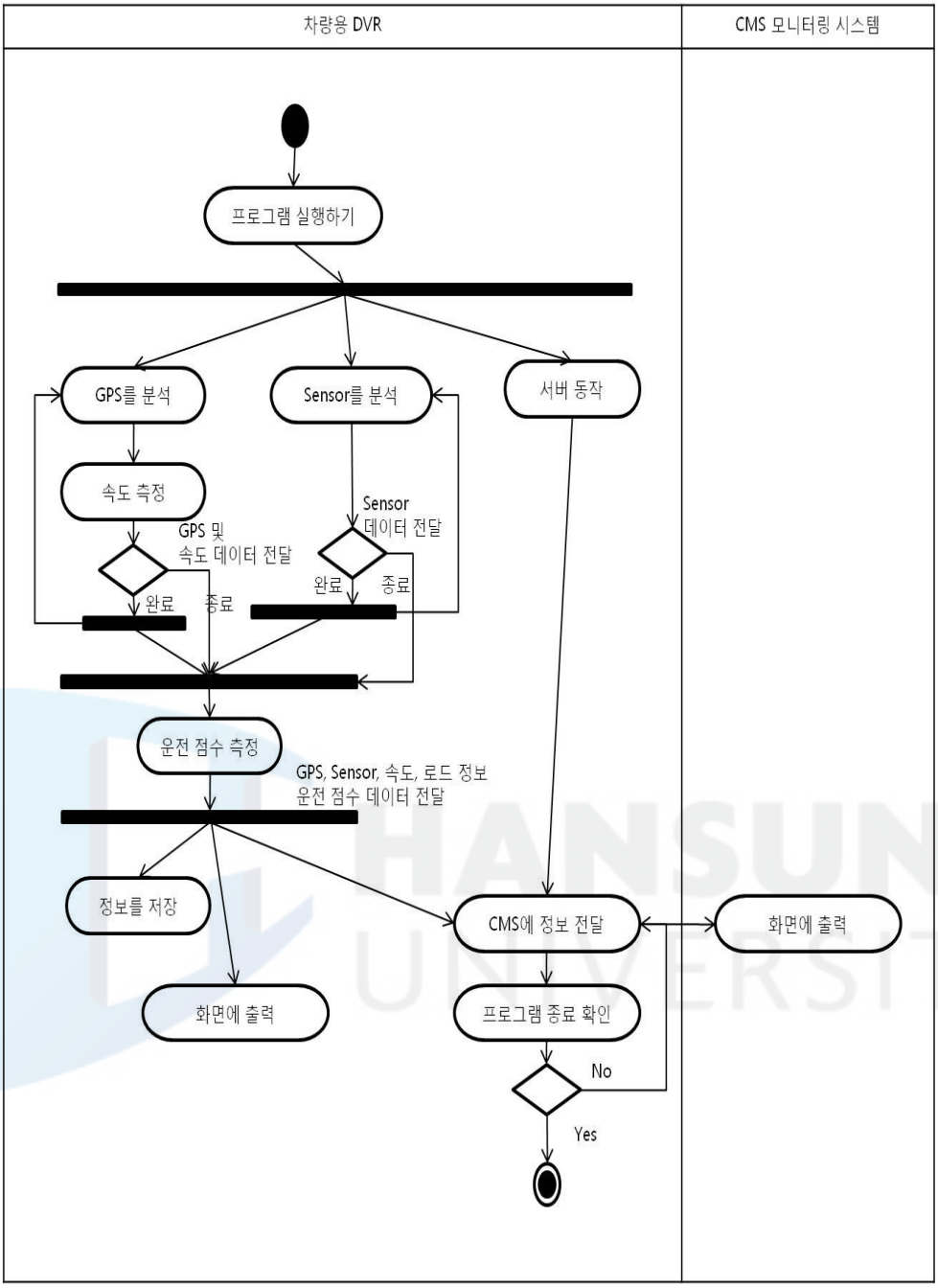


그림 4.2 차량용 DVR 활동 다이어그램

4.3.2 차량용 DVR 모듈

가. GPS 모듈

Serial이나 bluetooth로 통신하여 NMEA code(시간, 위치, 방위 등의 정보를 전송하기 위한 규격)를 통하여 우리가 원하는 GPS 정보를 가져온다. 또한, 분석된 GPS 정보 중 위도, 경도 및 시간을 가지고 차량 속도를 계산한다.

NMEA 라고 주로 불리는 NMEA 0183은 시간, 위치, 방위 등의 정보를 전송하기 위한 규격이다. NMEA 0183은 미국의 The National Marine Electronics Association에서 정의해 놓았다. 이 데이터들은 주로 gyrocompass, GPS, 나침반, 관성항법장치(INS)에 사용된다. NMEA 0183은 ASCII와 직렬 방식의 통신을 사용한다.

나. Sensor 모듈

윈도우즈에 phidget21 driver를 설치하여 그림과 4.3과 같은 모션 센서로부터 얻어진 raw 데이터를 이동평균 필터링을 통하여 3축 가속도 값을 얻어오는 모듈이다.

그림 4.3 PhidgetSpatial

다. 운전 점수 판단 모듈

선행 연구인 「위험운전 유형 분류 및 데이터 로거개발, 한국 ITS 학회지 제 7권, 제 3호, 2008, 6」 논문에서 분류하고 있는 위험운전 유형을 본 논문에 적용하였으며, 임계값을 도출하기 위하여 적용된 위험운전 유형은 표 4.1과 같다.

표 4.1 위험운전 유형

속도위반	직진구간 과속
	선회구간 과속
가속	급출발
	급가속
감속	급가속
	급감속
회전	급차선 변경
	연속적인 급차선변경

논문에서는 GPS의 위도, 경도 값을 가지고 로드 정보를 수집하고 MEMS (Micro Electro Mechanical Systems) 센서의 가속도 값과 속도 값을 수집하고 표 4.1과 같은 위험 운전 유형에 따라 점수화 하였다. 즉, 이들 상황에 유형별 가중치를 주어 운전자에게 정량적으로 이들 상황을 인지할 수 있는 난폭운전 측정 시스템 등 운전 안정성 관리 모듈을 개발하였다.

라. 통신 모듈 (서버)

CMS 모니터링 시스템(클라이언트)과 통신을 하면서 실시간 정보를 요청하면 현재 GPS와 Sensor 정보들을 통합하여 전달한다. 또한 이전 정보를 요청하면 원하는 시간에 맞추어 이전 데이터 정보를 전달하는데 전달하는 과정에서 정지, 빠르게, 느리게 및 일시 정지 등의 제어 명령들을 CMS 모니터링 시스템에서 전달받아 전달하는 속도를 조절하면서 데이터를 전송하는 모듈이다.

마. Display Module

그림 4.4와 같이 경도, 위도, 속도, 점수 및 날짜를 화면에 실시간 출력해 주고 오류메시지가 발생하였을 경우 리스트 박스를 통하여 오류 메시지를 출력해 준다.

그림 4.4 차량 DVR 출력 화면

마. 파일 관리 모듈

시간 단위로 Sensor 데이터 값과 GPS 데이터 값을 따로 분리하여 텍스트 파일로 저장한다. Sensor 데이터는 운전 점수, 가속도 센서 x축의 벡터 값, 가속도 센서 y축의 벡터 값, 가속도 센서 z축의 벡터 값, 최근 10초 동안의 가속도 센서 x축의 벡터 값, 최근 10초 동안의 가속도 센서 y축의 벡터 값, 최근 10초 동안의 가속도 센서 z축의 벡터 값, 최근 1분 동안의 가속도 센서 x축의 벡터 값, 최근 1분 동안의 가속도 센서 y축의 벡터 값, 최근 1분 동안의 가속도 센서 z축의 벡터 값들을 GPS는 위도, 경도, 현재 속도, 최근 10초 동안의 속도, 최근 1분 동안의 속도 값들을 저장하고 사용자가 원하는 시간의 데이터 정보를 읽어 올 수 있는 모듈이다.

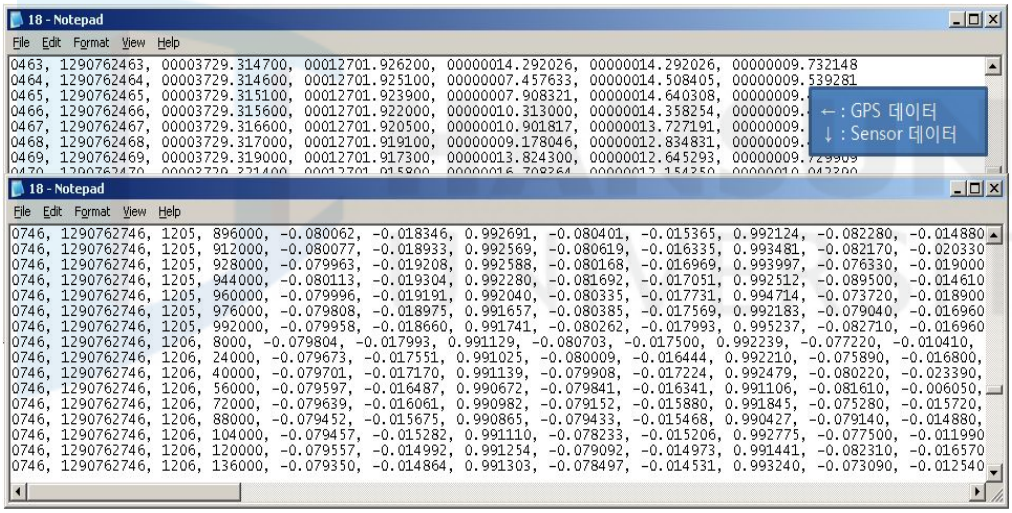


그림 4.5 GPS 및 가속센서 샘플

4.3.3 필터링

실험적인 환경에서의 가속도 그래프의 움직임을 보면 특징적인 움직임을 알 수 있을 만큼 깔끔하게 나온다. 하지만 실제로 장비가 운용되는 주행 중인 차량의 내부임을 감안하면, 센서 값에 잡음이 섞여서 들어올 것이란 사실을 쉽게 알 수 있다.

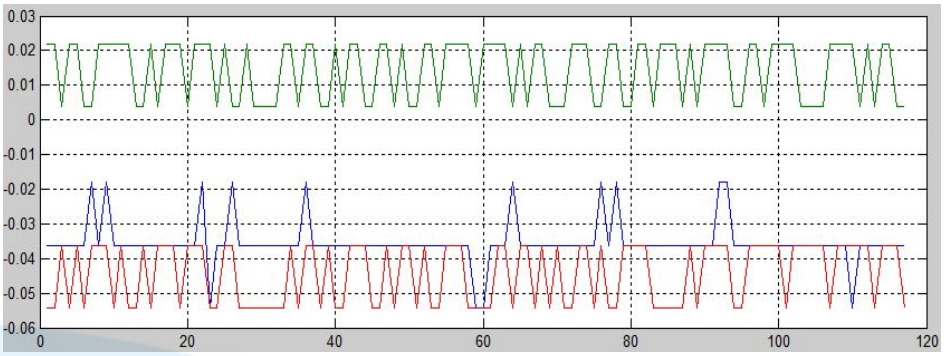


그림 4.6 움직이지 않았을 경우의 센서 값

그림 4.6을 보면, 입력되는 센서들의 값이 일정한 값으로 빠르게 진동하는 것을 볼 수 있다. 빨간 선은 녹색 그래프의 평균값이 약 0.015임을 나타내고 있다. 따라서 동적으로 신호를 안정 시켜주는 적응적 필터링을 통하여 이 문제를 해결하였다.

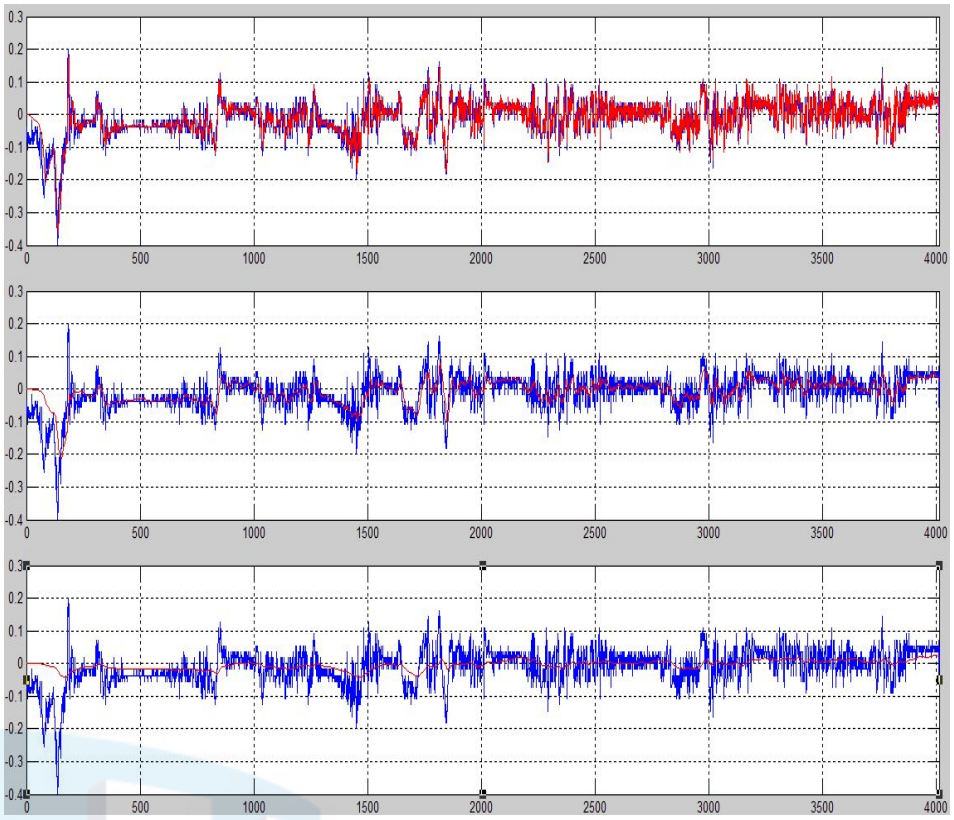


그림 4.7 LMS 필터링 결과

(파란색 : 원래의 가속도 데이터, 빨간색 : 필터링 된 값)

그림 4.7은 위에서 부터 MU값을 0.1, 0.01, 0.001로 분 결과이다. MU값은 적은 값일수록 오차 범위가 작아지지만, 클수록 원하는 값에 빠르게 수렴할 수 있다. 또한, Leaky LMS 필터를 적용하여 보았다.

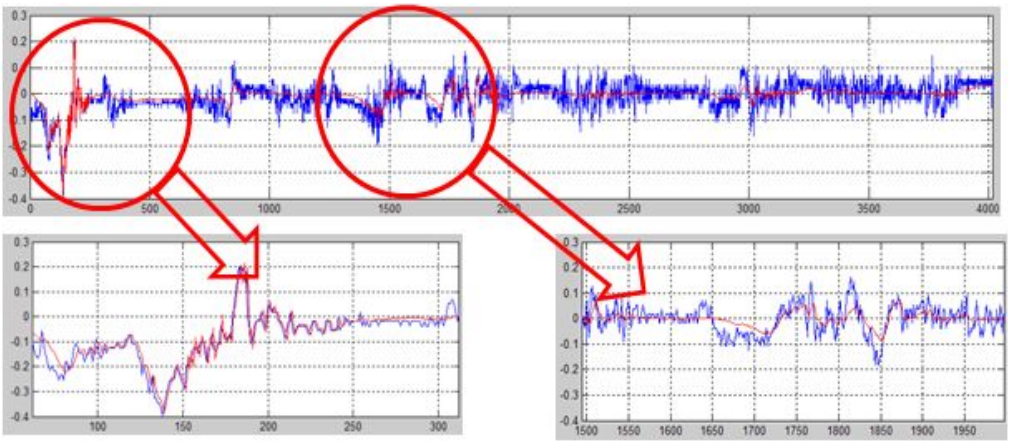


그림 4.8 Leaky LMS 필터링 결과

그러고 나서 이동평균 필터를 이용하여 값을 걸러보았더니 그림 4.9 처럼 좋은 결과가 나왔다.

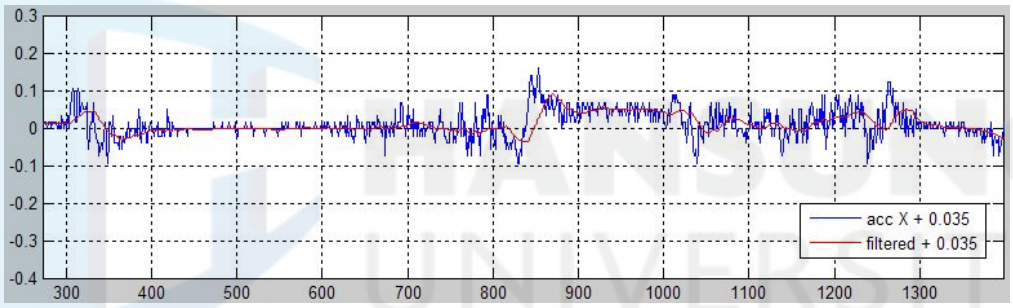


그림 4.9 이동평균 필터를 이용한 필터링 결과

4.3.4 드라이브 점수 측정 알고리즘

드라이브 점수 측정 알고리즘은 표 4.1 에 정리된 위험 운전의 특징을 파악하여 위험운전 유형을 정확하게 판별할 수 있어야 한다. 이러한 위험운전 특징을 파악하기 위하여 실차 실험을 통해 각 위험운전 상황에서의 차량 거동을 분석하였다.

본 논문에서 개발된 시스템에서는 운전 판단을 위해 차량에 장착된 가속도 센서와 GPS를 사용한다. 가속도 센서와 GPS로부터 취득한 데이터의 유형, 주기를 가지고 실시간으로 판단한다. 또한 개발된 드라이브 점수 측정 알고리즘은 여러 차례 실차실험을 통해 상세 파라미터 값을 조정하였고, 알고리즘 타당성 및 유효성을 확인하였다.

가. 선회 구간 및 직진 구간 판단

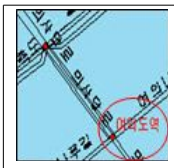
현재 우리나라 교통망 [노드(Node)/링크(Link)] 체계는 지능형교통체계[이하 "ITS(Intelligent Transportation Systems)"] 사업주체 별로 각기 개발되어 교통정보의 상호교환이 원활하게 이루어지지 못하고 있는 실정이었으나 국토해양부에서 교통정보 산업과 관련된 많은 업계에서는 교통정보의 원활한 교환을 위한 전국 교통망에 대해 단일화된 ID체계를 적용한 표준교통망 DB 구축이 하였다.

그림 4.10 노드 / 링크 개념

지능형교통체계 표준 노드/링크를 보면 그림 4.10과 같다. 노드는 차량이 도로를 주행함에 있어 속도의 변화가 발생되는 곳을 표현한 곳이고 노드유형으로는 교차로, 교량시종점, 고가도로시종점, 도로의 시종점, 지하차도시종점, 터널시종점, 행정경계, IC/JC 등이 있다.

링크는 속도변화 발생점이 노드와 노드를 연결하는 선을 의미하며 실세계에서의 도로이고 링크 유형으로는 도로, 교량, 고가도로, 지하차도, 터널 등이 있다.

표 4.2 노드/링크 구성

<table>
<tr>
<td>  </td>
<td>
<table>
<tr><th>필드명</th><th>속성명</th></tr>
<tr><td>NODE_ID</td><td>노드 ID</td></tr>
<tr><td>NODE_TYPE</td><td>노드유형</td></tr>
<tr><td>NODE_NAME</td><td>교차로명칭</td></tr>
<tr><td>TURN_P</td><td>회전제한유무</td></tr>
<tr><td>REMARK</td><td>비고</td></tr>
</table>
</td>
<td>☞ 노드
링크를 구분하는 점(링크 시작점,링크 종료점)으로 표준 로드/링크 구축 운영 지침에 따라 도로교차점,도로 시종점,교통통제점, 도로 구조 변환점,행정구역 변환점, 도로 운영 변환점, 교통 진출입점. 그외에 ITS사업 주체가 필요에 따라 정하는 지점

☞ 부가정보(회전제한정보)
회전제한이 존재하는 노드에 한해 회전제한 상세 정보를 입력한 테이블</td>
</tr>
<tr>
<td>  </td>
<td>
<table>
<tr><th>필드명</th><th>속성명</th></tr>
<tr><td>LINK_ID</td><td>링크 ID</td></tr>
<tr><td>F_NODE</td><td>시작노드 ID</td></tr>
<tr><td>T_NODE</td><td>종료노드 ID</td></tr>
<tr><td>ROAD_USE</td><td>도로사용여부</td></tr>
<tr><td>LANES</td><td>차로수</td></tr>
<tr><td>ROAD_RANK</td><td>도로등급</td></tr>
<tr><td>ROAD_TYPE</td><td>도로유형</td></tr>
<tr><td>ROAD_NO</td><td>도로번호</td></tr>
<tr><td>ROAD_NAME</td><td>도로명</td></tr>
<tr><td>MULTI_LINK</td><td>중용구간여부</td></tr>
<tr><td>CONNECT</td><td>연결로코드</td></tr>
<tr><td>MAX_SPD</td><td>최고제한속도</td></tr>
<tr><td>REST_VEH</td><td>통과제한차량</td></tr>
<tr><td>REST_W</td><td>통과제한하중</td></tr>
<tr><td>REST_H</td><td>통과제한높이</td></tr>
<tr><td>REMARK</td><td>비고</td></tr>
</table>
</td>
<td>☞ 링크
노드와 노드를 이은 도로중심선을 방향별로 일정간격 이격시켜 생성한 선으로서 실제 도로구간에 대한 정보를 담게됨

☞ 링크부가정보(중용구간정보)
링크가 중복노선으로 사용되어지는 중용구간인 경우에 한해 해당하는 중용구간 정보를 입력한 테이블</td>
</tr>
<tr>
<td>  </td>
<td colspan="2">- 행정권역
표준 노드/링크 구축 운영지침이 정의한 시군구별 권역코드 및 행정경계 데이터</td>
</tr>
</table>

도로에는 두 가지로 분류할 수 있는데 직진구간과 선회구간으로 분류할 수 있으며 선회구간을 판단하는 조건으로는 그림 4.11처럼 두 가지 경우가 있다.

첫 번째 경우는 링크와 링크가 변하는 시점을 선회 구간 시 위험운전 유형으로 판단한다. 또한 두 번째의 경우는 링크 정보 중에 정점들의 좌표들을 통하여 정점이 변화는 구간에 만약 방위각이 50이상일 경우에 선회 구간 시 위험운전 유형으로 판단한다. 그 밖에 구간을 직진 구간 시 위험운전 유형으로 판단한다.

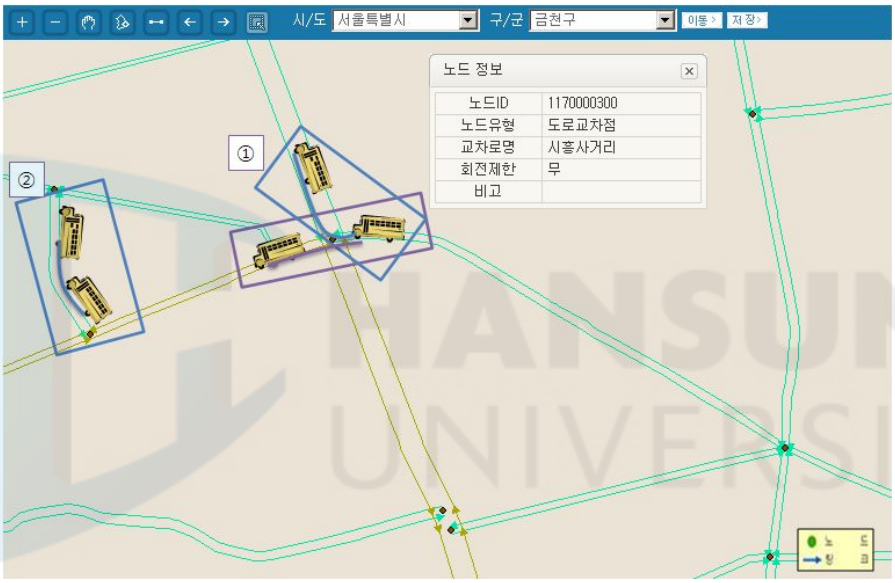


그림 4.11 직진 및 선회 구간 차량 움직임 표시

나. 위험운전 유형

운전 유형 가운데 적용 가능한 8가지 위험 운전 유형을 가지고 표 4.3과 같이 직진 구간의 경우와 선회 구간의 경우로 나누어서 적용하였으면 직진 구간 시 위험운전 유형은 선회구간을 제외한 7가지 위험운전 유형을 선회 구간 시 위험운전 유형은 회전을 제외한 6가지 위험운전 유형을 가지고 판단하였다.

표 4.3 직진 및 선회 구간 시 위험운전 유형

		위험운전 유형	
		직진 구간 시	선회 구간 시
속도위반	직진구간 과속	O	O
	선회구간 과속	X	O
가속	급출발	O	O
	급가속	O	O
감속	급가속	O	O
	급감속	O	O
회전	급차선 변경	O	X
	연속적인 급차선변경	O	X

다. 직진구간 과속

현재 우리나라에서의 제한속도는 법으로 정해진 속도이며, 구간에서의 위험운전 판단 기준은 지능형교통체계 표준 노드/링크 중 링크의 부가정보 중 MAX_SPD(최고속도) 필드 값을 근거로 판단하였다.

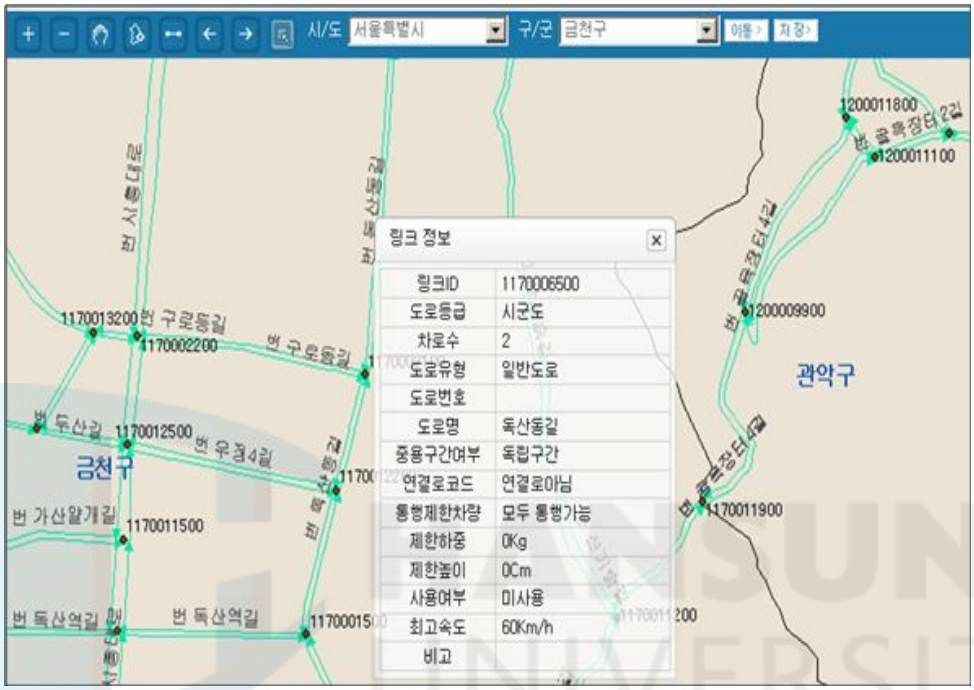


그림 4.12 지능형 교통체계 (표준노드 / 링크 갱신현황)

만약 정의된 도로 위에 없을 경우에는 표 4.4와 같이 교통법규에 정한 제한속도를 근거로 직진구간 과속을 판단하였다.

표 4.4 교통법규에 따른 제한속도

도로	차선별	제한속도
일반도로	편도 2차선 미만	최고 60km/h
	편도 2차선 이상	최고 80km/h
자동차 전용도로		최고 30km/h
		최고 90km/h
고속도로	편도2차선 이상	최고 50km/h
		최고 100km/h
		최고 80km/h (적재중량 1.5톤 초과 화물자동차, 특수자동차, 건설기계)
	편도1차선 이상	최저 40km/h
		최저 80km/h
	중부선	최저 60km/h
		최고 110km/h
		최고 90km/h (적재중량 1.5톤 초과 화물자동차, 특수자동차, 건설기계)

다. 선회구간 과속

선회구간에서는 위험운전 판단기준은 도로포장구조설계지침서(AASHTO)에 정의된 차량의 선회반경에 따른 속도 관계식을 근거로 제시하였다.

$$V = \sqrt{127R(e+f)}$$

여기에서, V : 설계속도 (km/h)
R : 선회반경 (m)
f : 횡방향 미끄럼 마찰계수
e : 편경사 (m/m)

그림 4.13 선회 반경에 따른 속도 계산식

그림 4.13의 속도-선회반경의 관계에 따라 선회반경 대비 안전속도를 보면 20m에서 약 28km/h, 40m에서 약 37km/h, 60m에서 약 43km/h, 80m에서 약 47km/h, 100m에서 약 52km/h로 분석되었다. 따라서 선회 반경이 커지면 그에 대한 안전속도도 커지는 것을 확인하였다. 선회 구간 동안의 거리를 통하여 선회반경을 구하고 선회구간 시 최고 속도와 비교하여 선회구간 과속을 판단하였다.

라. 급출발

급출발에 대한 위험운전 판단기준은 버스가 출발할 때 입석 승객이 불쾌감을 느낄 수 있는 경우를 위험운전의 기준으로 제시하고자 하며, 뉴턴의 제1법칙인 관성의 법칙을 응용할 수 있다.

관성의 법칙은 외부에서 작용하는 힘의 합력이 0일 때 정지해 있는 물체는 계속 정지해 있고, 운동하던 물체는 계속 등속도 운동을 한다는 자연법칙이다.

$$\sum_i \vec{F}_i = 0 \Rightarrow \vec{a} = 0$$

그림 4.14 정지 또는 등속도 운동

관성력은 관성 때문에 생기는 힘으로 실제 힘은 아니며, 가속도 운동하는 경우에 느낄 수 있는 힘이다. 따라서 버스가 출발하면 발과 버스 바닥 사이의 마찰력 때문에 하체는 차량과 함께 움직이기 시작한다. 그러나 상체는 관성에 의해 원래의 자리에 있기 때문에 몸이 뒤로 쏠리고, 즉 차량 안에서 보면 마치 승객들을 뒤로 잡아당기는 힘이 있는 것처럼 보이며, 이것을 바로 관성력이라 한다. 관성력은 차량가속도에 의한 힘과 크기가 같고, 방향은 차량의 움직임 반대로 작용한다.

따라서 차량의 급출발로 인해 가속도가 순간적으로 크게 작용하면 그만큼 관성력이 크게 작용하여 차량에 탑승한 승객에게 불쾌감 또는 위험한 상황을 초래할 수 있다. 그림 4.15는 급출발하였을 때의

x축 가속도를 측정한 값이며 급출발 판단은 속도가 0km/h에서 40km/h까지 속도가 급격하게 증가될 경우 (5초안에 30km/h를 도달하였을 경우) 센서 값을 보고 급출발인지 판단하였다.

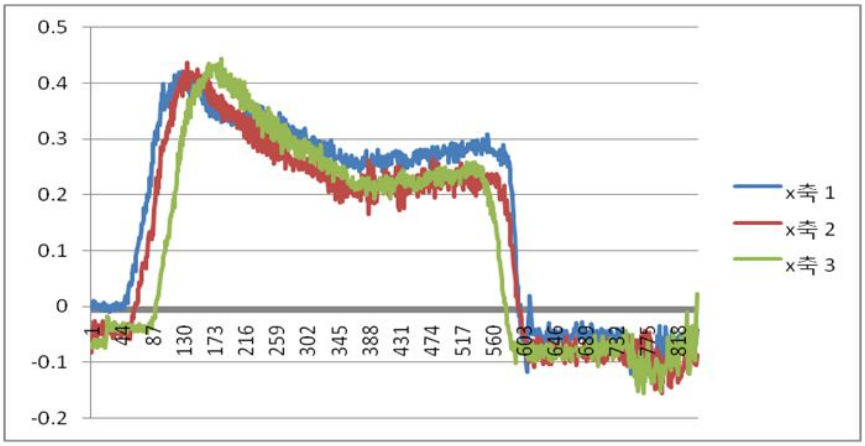


그림 4.15 0km/h~40km/h 급출발

마. 급가속

급가속에 대한 위험운전 판단기준은 급출발과 비교했을 때 일정한 속도로 주행 중 최대가속으로 속도를 증가시킬 때 차량의 진행방향의 가속도에 대한 실차 실험 결과를 위험운전 판단기준으로 한다.

정지한 상태에서 출발하는 차량의 가속도 변화량은 주행 중 속도를 증가시킬 때의 가속도 변화량보다 큰 값을 지닌다. 따라서 탑승한 승객이 불쾌감을 초래할 수 있는 값은 그림 4.16처럼 주행 중에 급가속은 센서 값의 변화량의 크기를 보고 판단한다.

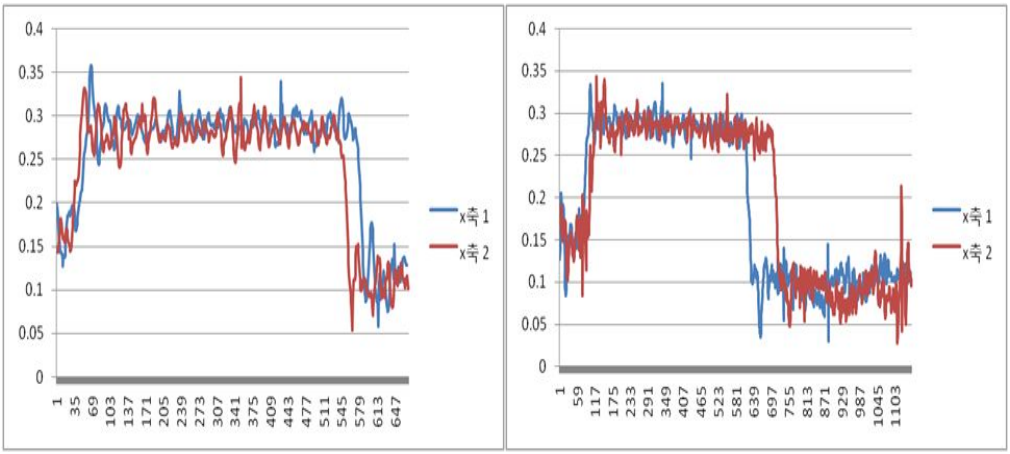


그림 4.16 60km/h~80km/h 급가속 및 80km/h~100km/h 급가속

또한 선회의 경우 선회 반경 관련 안전속도에 따른 실차 실험결과 정상 선회 시 횡 방향 가속도 결과가 비슷한 값으로 나오는 것으로 확인할 수 있다. 따라서 선회 구간 시 속도가 증가 중일 때 횡가속도의 변화량을 보고 급가속을 판단한다.

표 4.5 선회 반경 관련 안전속도 및 횡가속도

선회 반경(m)	속도(km/h)		횡가속도(G)
20	28	➡	0.22
40	37		0.27
60	43		0.24
80	47		0.22
100	52		0.22

바. 급정지

급정지에 대한 위험운전 판단기준은 버스가 갑자기 정차할 때 입석 승객이 쓰러질 수 있는 경우를 위험운전의 기준으로 제시하고자 한다. 급출발에서 관성력에 대해 기술하였듯이 급정지나 급감속의 경우는 급출발과 급가속의 원리와 그 반대의 개념으로 생각하면 된다. 모든 물체는 자신의 운동 상태를 그대로 유지하려는 성질이 있으며, 정지한 물체는 계속 정지해 있으려 하고(급출발, 급가속에 해당), 운동하는 물체는 원래의 속력과 방향을 그대로 유지하려고 한다. (급정지, 급감속에 해당)

차량이 출발할 때와 반대로 차량이 멈출 때 하체는 힘을 받아 멈추지만 상체는 계속 앞으로 가기 때문에 앞으로 쏠리는 현상이 발생한다. 따라서 차량이 급정지로 인해 감속도가 순간적으로 크게 작용하면 그만큼 관성력이 크게 작용하여 차량에 탑승한 승객에게 불쾌감 또는 위험한 상황을 초래할 수 있다.

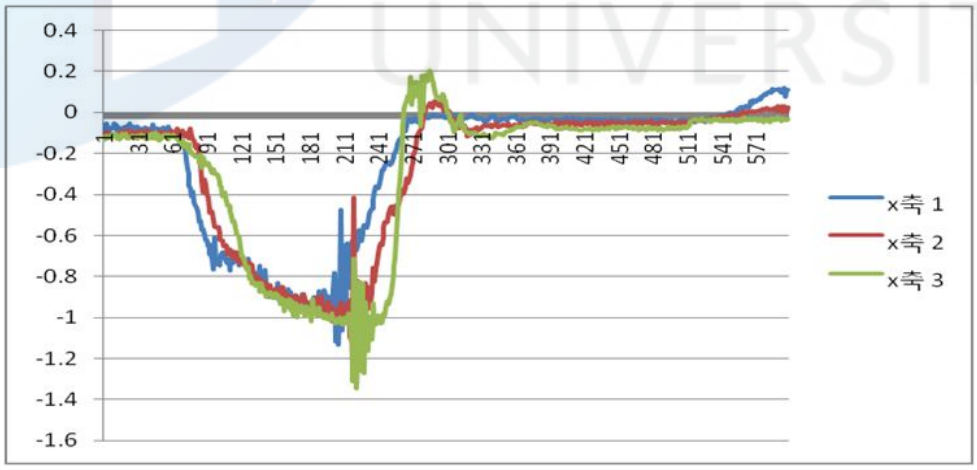


그림 4.17 40km/h~0km/h 급정지

그림 4.17은 급정지하였을 대의 x축 가속도를 측정한 값이며 급정지 판단은 속도가 40km/h에서 0km/h까지 속도가 급격하게 감소하였을 경우(5초안에 30km/h에서 속도가 감소하여 정지하였을 경우) 센서 값을 보고 급정지인지 판단하였다.

사. 급감속

급감속에 대한 위험 운전 판단기준은 급정지와 비교했을 때 일정한 속도로 주행 중 차량의 속도가 갑작스럽게 줄어든 운전행태로 차량의 진행방향의 가속도에 대한 실차 실험 결과를 위험운전 판단기준으로 한다.

급정지와 유사한 상황이지만 차량이 완전히 정지하지 않고 주행 중 차량의 속도가 갑작스럽게 감소하는 상황으로 급정지의 가속도 변화량이 급감속의 가속도 변화량이 큰 값을 지닌다. 따라서 그림 4.18처럼 센서 값의 변화량의 크기를 보고 판단한다. 또한 선회 구간 시 속도가 감소 중일 때 횡가속도의 변화량을 보고 급가속을 판단한다.

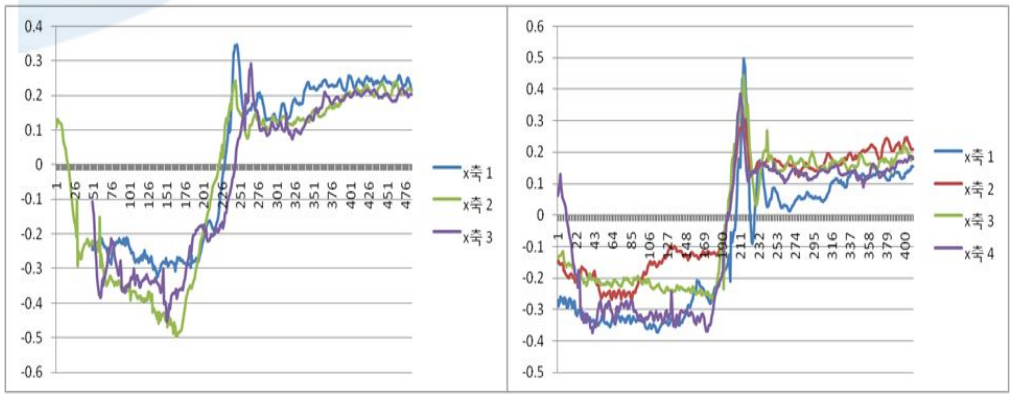


그림 4.18 80km/h~60km/h 급감속 및 100km/h~80km/h 급감속

아. 급차선 변경, 연속적인 급차선 변경

급차선 변경은 주행속도에 비해 조향 핸들을 급격하게 조작하는 운전행태를 의미한다. 급차선 변경 시 노면의 상태나 도로 굴곡에 따라 차량전복의 위험까지 처할 수 있다. 따라서 직진 구간 시 횡가속도에 대한 실차 실험 결과를 위험운전 판단기준으로 한다.

직진 구간에서의 횡가속도 변화량은 그림 4.19처럼 각 20km/h, 30km/h, 40km/h, 50km/h에서 급차선 변경하면서 주행할 때 차선 변화를 보면 Sine파의 주기로 나타내면 주기가 짧거나 변화량이 클 경우 속도대비 조향조작을 급하게 한 것으로 판단한다.

연속적인 급차선 변경은 급차선 변경의 위험운전 판단원리와 같으며, 위험운전 판단기준은 연속적인 Sine 주기의 시간과 횡가속도의 변화량이다. 차선 변경하는 동안의 시간 측정은 Sine 주기가 발생하고 일정 시간 이내에 연속적인 Sine 주기가 발생하였을 경우가 해당된다.

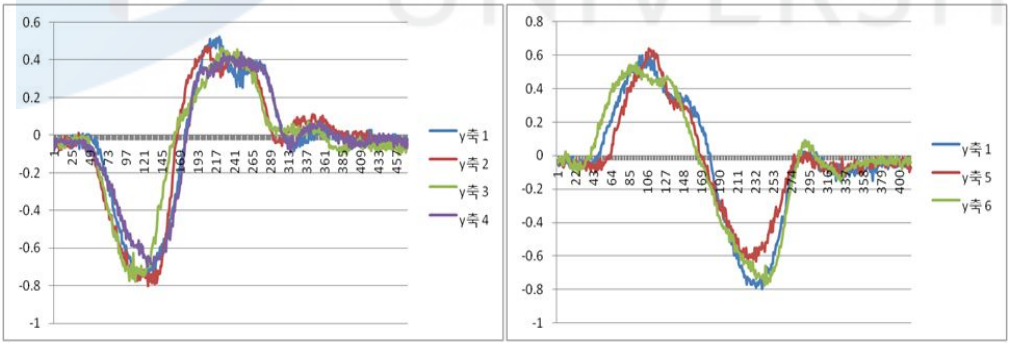


그림 4.19 급차선 변경 (오른쪽 / 왼쪽)

자. 위험 운전 유형에 대한 가중치

본 논문에서는 위험운전 유형에 따른 100점 만점에 경고 메시지를 제공하기 위한 기준을 선정하기 위하여 운전을 할 수 있는 일반인을 대상으로 실차 실험을 진행하였다. 진행한 결과 표 4.6과 같은 임계값과 가중치를 측정할 수 있었으며 향후 버스차량 승객들의 데이터를 이용하여 수정・보완할 예정이다. 또한, 각 임계값에 초과하는 값들에 따른 차등 점수를 부과하였다. 속도의 경우에는 1km/h마다 1점씩 추가 감점하였고 가속도 센서는 0.01마다 추가로 감점하였다. 이를 통하여 종합적인 위험운전 점수를 측정하여 운전자와의 현실적인 실시간 교감을 할 수 있게 만들었다.

표 4.6 위험운전 유형에 대한 가중치

위험운전 유형		판단변수	임계값	선회 가중치	직진 가중치
속도 위반	직진구간과속	차속 (GPS)	법규에 준함	70%	20%
	선회구간과속	GPS	표 4.5를 참조	70%	20%
가속	급출발	종가속도	0.3G	20%	20%
	급가속	종가속도	0.3G		
감속	급정지	종가속도	0.3G	20%	20%
	급감속	종가속도	0.3G		
회전	급차선 변경	주기 및 주파수	5.0초	None	20%
		횡가속도	0.4G or -0.4G		
	연속적인 급차선 변경	주기 및 주파수	10.0초	None	
		횡가속도	0.4G or -0.4G		

※ 단 선회 가중치는 가속 + 감속 포함 30%, 직진구간과속 + 선회구간과속 70%이고 직진 가중치는 가속 + 감속 포함 30%이다.

4.4 CMS 중앙 모니터링 시스템 구현

4.4.1 CMS 중앙 모니터링 시스템 활동 다이어그램

차량용 DVR(서버)와 통신을 하면서 요청한 실시간 정보와 이전 운전 정보를 전달받아 구글 맵 Open Api를 사용하여 화면에 표시하는 과정을 그림 4.20에서 보여준다.

수행 과정을 보면 프로그램이 시작되면 IP 및 포트를 입력하여 차량용 DVR(서버)와 연결하고 실시간 또는 이전 정보를 요청하여 차량용 DVR로부터 주기적으로 운전 정보인 위치 정보, 운전 점수 및 속도를 전달받아 구글 맵 Open Api를 사용하여 화면에 표시해 준다.

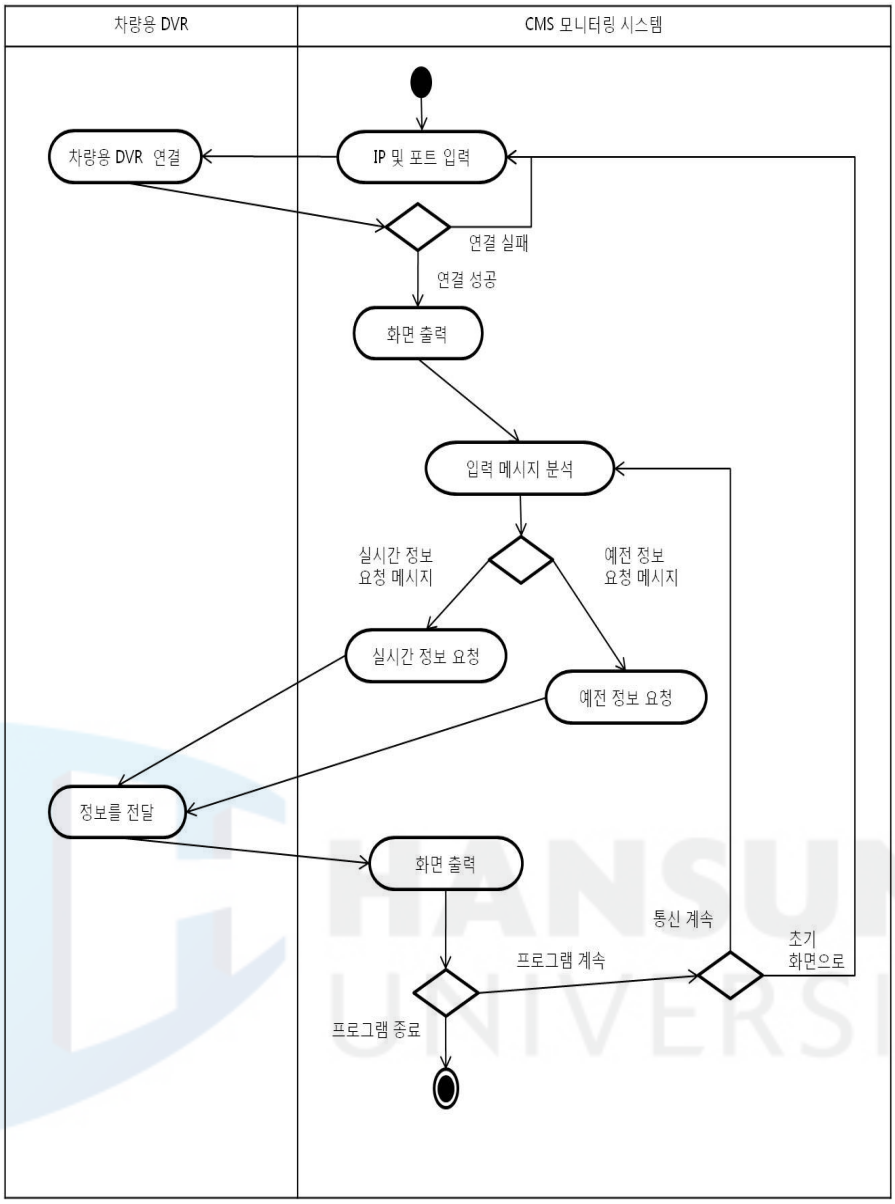


그림 4.20 중앙 모니터링 시스템 활동 다이어그램

4.4.2 차량용 DVR 모듈

가. 통신 모듈 (클라이언트)

차량용 DVR(서버)와 통신을 하면서 실시간 정보와 이전 운전 정보를 요청할 수 있다. 또한 차량용 DVR로부터 운전정보를 전달받아 Google api를 사용하여 화면에 출격해 주는 Display 모듈에 데이터를 전달해 준다. 그리고 이전 운전 정보를 출력하는 과정에서 재생, 정지, 일시정지, 빠르게 및 느리게 등의 제어 명령어를 보낼 수 있어 차량 DVR에서 전송 받는 속도를 조절할 수 있다.

나. Display 모듈

구글 맵 Opne api를 사용하여 Map을 확보하고 Qt 툴을 이용하여 GUI를 구현하였다. 다음 그림 4.21은 CMS 모니터링 시스템에서 차량 DVR(서버)에 연결되어 운전 정보를 받아 화면에 현재 위치와 운행 점수를 표시한 예이며 만약 맵 위에 보이는 차량을 클릭하게 되면 우측상단에 보이는 것처럼 시간과 점수 그리고 속도를 보여준다. 그리고 구글 맵에 연동되지 않을 경우를 대비하여 에디트 박스에 위치 정보인 경도, 위도와 점수 및 시간을 보여준다.

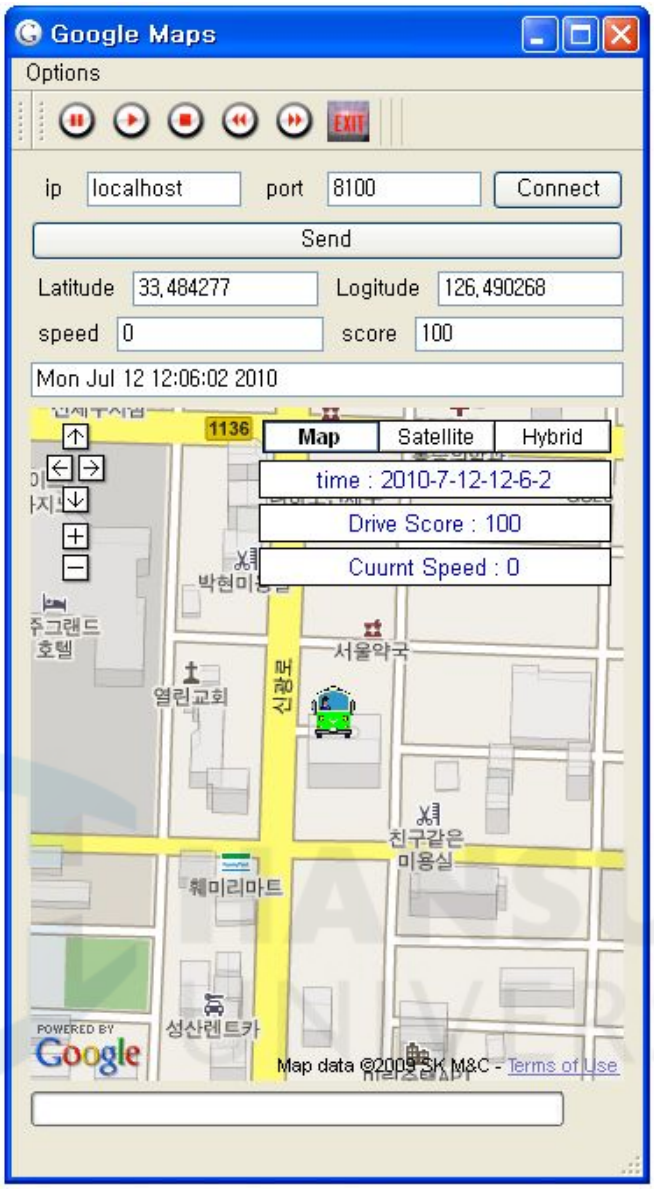


그림 4.21 CMS 모니터링 시스템 실시간 영상화면

그리고 그림 4.22처럼 이전 운전 정보들을 검색하여 확인하고자 할 때 시작과 종료 시간 및 시간 주기를 입력하여 확인할 수 있으면 이전 운전 정보들을 보면서 제어 명령어를 통해 재생 속도 조절을 할 수 있다.

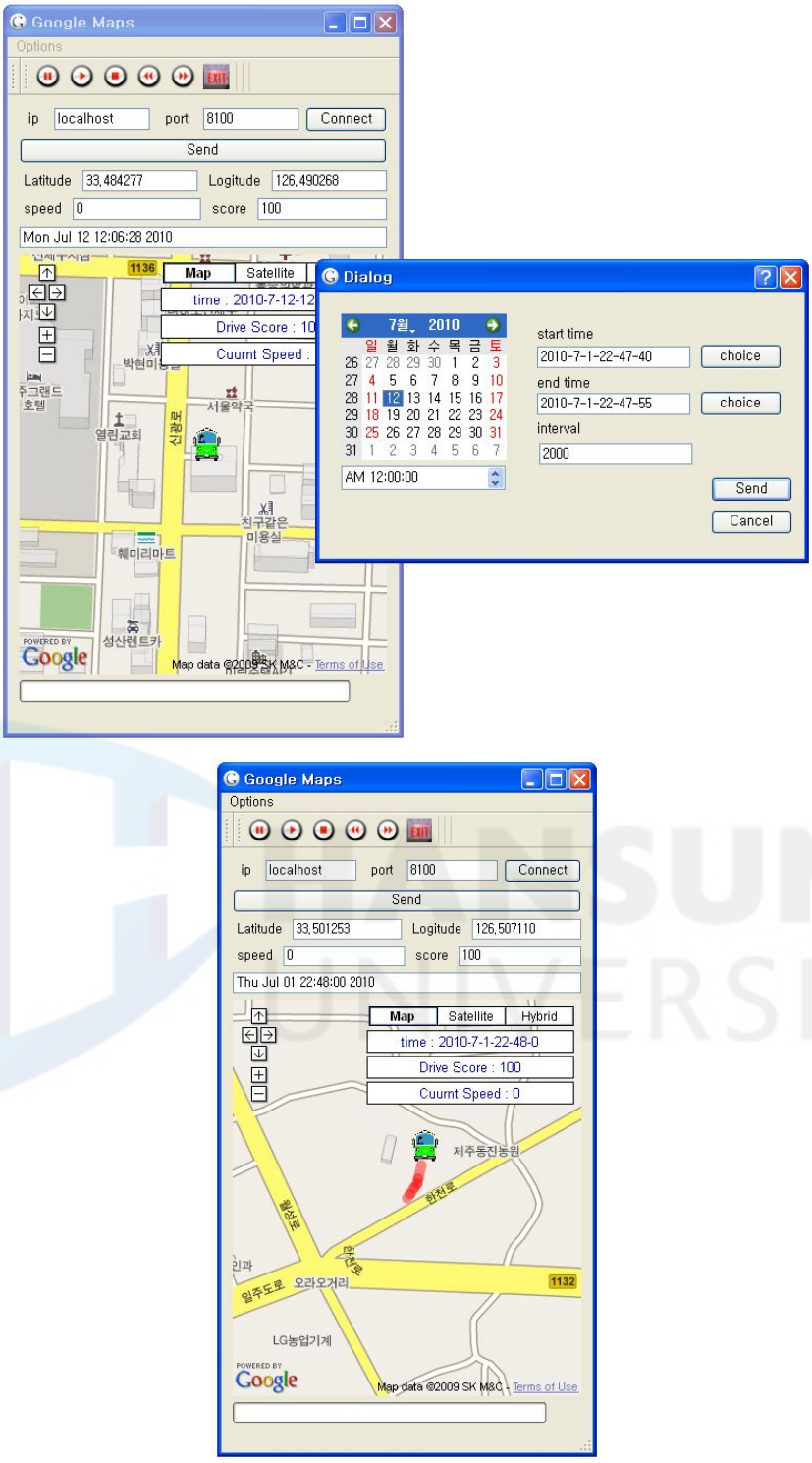


그림 4.22 이전 운전 정보 검색 및 출력

제 5 장 GPS와 가속도 센서 기반 정숙 운전 유도 시스템 파급효과 및 활용 방안

5.1 연료비 절감 효과

운행 기록기의 데이터는 보다 안전하고 경제적인 운전을 위한 교육 자료로도 쓰일 수 있다. 제동, 가속, 감속 등에 관한 운전 특성이 양적으로 측정되므로 다른 운전자와의 비교를 통해 안전의식을 높이고 급가속이나 급제동을 줄이는 효과를 얻게 되는 것이다. 이로써 사고 횟수뿐만 아니라 연료비도 줄일 수 있다.

일본의 한 조사에 따르면 기록기를 장착한 차량은 그렇지 않은 차량에 비해 10%가량 연료비가 줄어드는 것으로 나타났다. 가감속이 잦을수록 연료비도 많이 들기 때문이다. 운행 기록기의 연료비 절감 효과가 실로 대단하다는 사실을 알 수 있는 대목이다.

5.2 승객의 승차감 향상

본 논문에서는 GPS와 가속도 센서를 통한 운행 안정성을 측정하여 디스플레이에 수치 표시하여, 운전자의 운행 정숙도를 승객이 확인할 수 있다. 이는 운전자가 승차감을 고려하여 정숙한, 안정성 높은 운전을 할 수 있도록 유도하여 대중교통 수단에서의 승차감 향상에 따른 편의성을 증가시킬 수 있다.

5.3 보험사에도 유리

GPS와 가속도 센서 기반 정숙 운전 유도 시스템을 장착한 차량은 그렇지 않은 차량에 비해 사고가 적고, 설령 사고가 발생한다 해도 사고 처리가 빠르다. 따라서 합의 과정에 들어가는 인력과 비용은 물론이고, 사고로 인한 보상금 지급 규모까지 줄일 수 있다. 그렇게 절감된 비용을 활용하여 다른 서비스를 소비자에게 제공할 수 있다.

5.4 자동차 메이커의 이점

저마다 안전과 환경을 강조하는 자동차 회사 역시 경쟁업체들 중에서 두드러져 보이기 위해 애쓴다는 점은 보험회사와 같다. 자동차 회사들이 환경을 위해 하이브리드 자동차 개발에 나서고 있는 것도 이런 맥락에서다.

안전 부문에서는 사고 예방이라는 측면이 매우 강조될 것이다. 바로 여기서 GPS와 가속도 센서 기반 정숙 운전 유도 시스템이 큰 역할을 할 수 있다. 일차적인 기능은 운행데이터 수집이다. 운행 기록기는 운전자가 급브레이크를 밟아야 하는 급박한 상황을 기록해주므로, 어느 상황에서 사고가 일어나는가를 파악하는 데 도움이 된다.

지금까지 자동차 회사들은 '사고가 발생했을 때의 탑승자 보호'에 중점을 두었던 게 사실이다. 에어백이나 사고 발생 시 적절히 찌그러지는 특수 차체 등이 대표적인 예이다. 하지만 관심의 초점은 점차 사고 예방으로 옮겨가고 있다. 사고를 아예 미리 막자는 뜻이다. GPS와 가속도 센서 기반 정숙 운전 유도 시스템이 운행정보 축적을 통해 사고를 일으키는 숨은 요인을 발견할 수 있을 것이다.

제 6 장 결론 및 향후 연구

위험 운전은 본인뿐만 아니라 동료들에게도 악습을 전파하는 특성이 있을 뿐만 아니라 사후 관리 부재 시 위험 운전 재발의 악순환으로 결국 사고를 유발한다.

본 논문에서는 가속도 센서, GPS를 이용하여 각 방향의 가속도, 차량의 방향, 속도 등 차량의 모든 움직임을 계산하고, 이전 연구에서 제시된 기준을 바탕으로 정숙 운전 지수를 계산한 뒤 구글맵 상에 차량의 정보와 함께 실시간으로 출력함으로써, 운전자의 운행 습관을 개선할 수 있도록 도와준다. 또 모든 데이터를 저장되어, 추후에 분석도 가능하다.

본 논문에서는 운전 점수 측정 알고리즘을 보완하기 위해서는 운전자나 탑승객의 주관적인 느낌도 중요하므로, 계산된 지수와 느낌과의 관계를 면밀히 비교하여 반영하여야 하며, 차량의 크기, 종류 따른 차이도 반영하기 위한 연구도 필요하다.

그리고 GPS와 가속도 센서 기반 정숙 운전 유도 시스템을 스마트폰에 내장되어 있는 모션센서와 GPS를 이용하여 정숙 운전 지수를 계산하는 어플을 만들 수 있다. 또한 스마트폰에서 계산된 운행 점수는 서버로 저장이 되고, 이 저장된 점수는 SNS 서비스를 통하여 트위터, 페이스 북 등으로 업데이트 할 수 있도록 하여 자신의 운전 실력을 지인들에게 뽐낼 수 있도록 하며, 자체적으로 랭킹사이트를 이용하여 같은 앱을 사용하는 사람들에게 자랑할 수 있다.

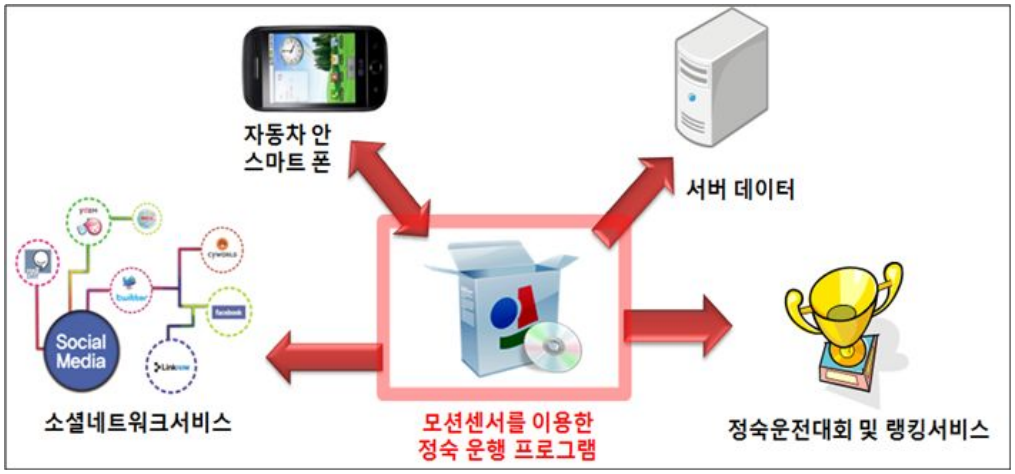


그림 6.1 GPS와 가속도 센서 기반 정숙 운전 유도 시스템 향후 연구 분야

【참고문헌】

국내문헌

오주택 외 1인, Vision 시스템을 이용한 위험운전 원인 분석 프로그램 개발에 관한 연구, 『한국ITS학회 논문지 제 8권』 제 6호, 2009년 12월

오주택 외 3인, 위험운전 유형 분류 및 데이터 로거개발, 『ITS학회 논문지 제 7권』 제 3호, 2008년 6월

_____, 위험운전 유형에 따른 임계값 개발, 『한국도로학회 논문집, 제 11권』 제 1호, 2009년 3월

이운성 외 1인, 안전운전 관리시스템 개발, 『한국자동차공학회 논문집 제 5권』, 2007년

한인환 외 1인, 차량용 블랙박스를 활용한 위험 운전 인지, 『대한교통학회지 제 25권』 제 5호, 2007년 10월

<관련 웹 사이트>

- 사이버 경찰청, http://police.go.kr/main/index_info.do
- 지능형교통체계 표준노드링크관리시스템, http://nodelink.its.go.kr/
- Google Maps JavaScript API V3, http://code.google.com/intl/ko-KR/apis/maps/documentation/javascript/

【부 록】

GPS와 센서 API

Contents

1. API Summary

API	
int	(void) 등록되어 있는 callback 함수와 좌표 배열을 취소한다.
int	(void) 등록되어 있는 callback 함수와 특정 센서 값을 취소한다.
int	(void) gps Thread을 안전하게 종료한다.
int	(void) 센서 Thread를 안전하게 종료한다.
int	(time_ttimeStamp) 입력된 해당 시간이전의 GPS 데이터를 파일에서 모두 삭제한다.
int	(time_ttimeStamp) 입력된 해당 시간이전의 센서 데이터를 파일에서 모두 삭제한다.
int	(void) gps를 초기화 하고 gps 데이터 값을 읽는 Thread를 시작한다.
int	(void) 센서를 초기화 하고 센서 값을 Thread를 시작한다.
int	(time_ttimeStamp,struct* buf, int count) 파일로부터 저장된 gps 데이터를 읽는다.
int	(time_ttimeStamp,struct* buf, int count) 파일로부터 저장된 센서 데이터를 읽는다.
int	(void (*func)(int pos_index), struct * pos, int count) 특정 좌표 리스트에 대한 callback 함수를 지정한다.
int	(void(*func)(void*),struct*base) 특정 센서 값에 대한 callback 함수를 지정한다.
int	(void) 감지되는 gps 데이터 값을 파일로 저장한다.
int	(void) 감지되는 센서 값을 파일로 저장한다.
int	(struct* gps_data_info) gps 데이터 정보를 반환한다.
int	(struct* sensor_data_info) 센서 데이터 정보를 반환한다.
int	(void) gps 데이터 파일을 닫는다.
int	(void) 센서 데이터 파일을 닫는다.

2. Data Structure

2.1 _GPS_DATA_INFO

```
#include "hs_sensor_gps.h"

struct _GPS_DATA_INFO {
  int saveInterval;
  time_t startTimeStamp;
  time_t endTimeStamp;
};
```

fields

saveInterval

GPS Data(시간, 좌표, 속도)를 저장하는 주기. (단위는 msec)

startTimeStamp

파일로 저장된 정보의 최초 시간.

endTimeStamp

파일로 저장된 정보의 가장 마지막 시간.

2.2 _GPS_DATA

```
#include "hs_sensor_gps.h"

struct _GPS_DATA {
  time_t timeStamp;
  struct pos;
  double currentSpeed;
  double avgSpeedTenSec;
  double avgSpeedMinute;
};
```

fields

timeStamp

저장되었던 시간. (4Byte UTC epoch sec형식)

pos

GPS 좌표.

currentSpeed

현재 속도. (단위는 km/h)

avgSpeedTenSec

최근 10초간 평균 속도. (단위는 km/h)

avgSpeedMinute;

최근 1분간 평균 속도. (단위는km/h)

2.3 _GPS_POSITION

```
#include "hs_sensor_gps.h"

struct _GPS_POSITION {
  double latitude;
  double longitude;
};
```

fields

latitude

위도. (ex. 3735.0079는 위도로서 37도 35.0079분을 뜻한다. 60분이 1도니까, 대략 37.5도가 된다. (37° 35.0079' = 37.583465° = 37° 35' 0.474"))

Longitude

경도. (ex. 12701.6446은 경도로서 127도 1.6446분을 뜻한다. 그러니까, 대략 127도가 된다. (127° 1.6446' = 127.02741° = 127° 1' 38.676"))

2.4 _SENSOR_DATA_INFO

```
#include "hs_sensor_gps.h"

struct  _SENSOR_DATA_INFO    {
   int   saveInterval;
   tiem_t    startTimeStamp;
   tiem_t       endTimeStamp;
};
```

fields

saveInterval

3축 가속도 센서를 저장하는 주기. (단위는 msec)

startTimeStamp

파일로 저장된 정보의 최초시간.

endTimeStamp

파일로 저장된 정보의 가장 마지막 시간.

2.5 _SENSOR_DATA

```
#include "hs_sensor_gps.h"

struct _SENSOR_DATA {
   time_t timeStamp;
   int score;
   struct currentAcc;
   struct avgAccTenSec;
   struct avgAccOneMin;
};
```

fields

timeStamp

데이터가 저장되었던 시간. (4Byte UTC epoch sec형식)

score

운전 점수를 저장합니다.

acc

가속도 센서의 x, y, z값. (단위는 중력가속도 (g) 이며, 1g는 9.8m/s^2이고 함수에서 센서가 없을 경우 acc의기본값을 0으로 설정)

avgAccTenSec;

최근 10초 동안 가속도 센서 값들의 평균.

avgAccOneMin;

최근 1분 동안 가속도 센서의 값들의 평균.

2.6 _3D_VECTOR

```
#include "hs_sensor_gps.h"

struct _3D_VECTOR {
        double x;
        double y;
        double z;
};
```

fields

x

x축의 벡터 값. (단위는 중력가속도 (g) 이며, 1g는 9.8m/s^2이다.)

y

y축의 벡터 값. (단위는 중력가속도 (g) 이며, 1g는 9.8m/s^2이다.)

z

z축의 벡터 값. (단위는 중력가속도 (g) 이며, 1g는 9.8m/s^2이다.)

2.7 전역변수

```
#include "hs_sensor_gps.h"

const char* gps_device = "/dev/com1";
const char* sensor_device = "/dev/com2";
const char* gpsSaveInterval = 1000;
const char* sensorSaveInterval = 1000;
const char* gps_data_save_path = "C:/GPS_DATA";
const char* sensor_data_save_path = "C:/SENSOR_DATA";
```

fields

gps_device

GPS 장치를 접근하기 위한 블루투스나 시리얼 포트.

sensor_device

가속도 센서를 접근하기 위한 장치 포트.

gpsSaveInterval

GPS 데이터 정보를 저장할 주기적인 주기. (단위는 msec)

sensorSaveInterval

센서 데이터를 저장할 주기적인 주기. (단위는 msec)

gps_data_save_path

GPS 데이터를 저장할 경로.

sensor_data_save_path

센서 데이터를 저장할 경로.

3. API Detail

3.1 cancel_gps_callback

Synopsis

```
#include "hs_sensor_gps.h"
int     cancel_gps_callback(void);
```

Description

등록되어 있는 callback 함수와 좌표 배열을 취소한다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_NO_CALLBACK : 등록된 callback 함수가 없다.

See Also

register_gps_callback()

3.2 cancel_sensor_callback

Synopsis

#include “hs_sensor_gps.h”

int cancel_sensor_callback(void);

Description

등록되어 있는 callback 함수와 특정 센서 값을 취소한다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_NO_CALLBACK : 등록된 callback 함수가 없다.

See Also

register_sensor_callback()

3.3 close_gps

Synopsis

#include “hs_sensor_gps.h”

int close_gps(*void*);

Description

현재 gps 스레드를 사용하고 있다면, gps 스레드를 종료한다. 종료하기 전에 데이터를 모두 파일로 저장하고 사용했던 자원을 모두 해제한다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_NOT_INITIALIZED: 초기화 된 적이 없다.

_HS_DEVICE_DISCONNECT_FAULT : 시리얼이나 블루투스의 연결 해제를 실패했다.

_HS_THREAD_EXIT_FAULT : Thread 종료를 실패했다.

See Also

init_gps()

3.4 close_sensor

Synopsis

```
#include  "hs_sensor_gps.h"
int       close_sensor(void);
```

Description

현재 센서 스레드를 사용하고 있다면, 센서 스레드를 종료한다. 종료하기 전에 데이터를 모두 파일로 저장하고 사용했던 자원을 모두 해제한다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_NOT_INITIALIZED: 초기화 된 적이 없다.
_HS_FILE_SAVE_FAULT : 파일 저장을 실패했다.
_HS_MEMORY_FREE_FAULT : 메모리 해제를 실패했다.
_HS_CALLBACK_FAULT : 콜백 함수의 종료를 실패했다.
_HS_THREAD_EXIT_FAULT : Thread 종료를 실패했다.

See Also

init_sensor()

3.5 delete_gps_data

Synopsis

#include “hs_sensor_gps.h”

int delete_gps_data(time_ttimeStamp);

Description

입력된 해당 시간이전의 GPS 데이터를 시간 단위로 파일에서 모두 삭제한다. 만약, 입력된 시간이 5월 1일 12시 10분이라면 12시 이전의 데이터 삭제되고 12시 00분부터 12시 10분까지의 데이터는 삭제되지 않는다.

Parameter

time_t timestamp : UTC epoch time 값.

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

See Also

read_gps_data()

3.6 delete_sensor_data

Synopsis

#include “hs_sensor_gps.h”

int delete_sensor_data(*time_t timeStamp*);

Description

입력된 해당 시간이전의 센서 데이터를 시간 단위로 파일에서 모두 삭제한다. 만약, 입력된 시간이 5월 1일 12시 10분이라면 12시 이전의 데이터 삭제되고 12시 00분부터 12시 10분까지의 데이터는 삭제되지 않는다.

Parameter

time_t timestamp : UTC epoch time 값.

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_INVAILD_TIME : 해당 시간의 데이터가 존재하지 않는다.

_HS_DELETION_OPERATION_FAULT : 삭제 연산을 실패한다.

_HS_TRIM_OPERATION_FAULT : 삭제 후 Trim 연산을 실패한다.

See Also

read_sensor_data()

3.7 init_gps

Synopsis

#include “hs_sensor_gps.h”

int **init_gps**(*void*);

Description

gps를 사용하기 위해 초기화 하고, gps 데이터 값을 주기적으로 읽는 Thread가 생성되지만 gps 정보를 파일로 저장 하지 않는다. 또한, 이 상태에서 함수에 timestamp를 0으로 하여 현재 값을 읽을 수 있다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_ALREADY_SERIAL_PORT_INITED : 이미 시리얼이 초기화되지 않았다.

_HS_DEVICE_HANDLE_CONNECT_FAULT : 디바이스의 연결이 실패했다.

_HS_DEVICE_HANDLE_CONFIGURATION_FAULT : 디바이스의 환경설정이 실패했다.

_HS_DEVICE_HANDLE_SETTING_COMMUNICATION_EVENT_MASK_ERROR : 디바이스 통신 이벤트 에러가 발생했다.

_HS_ALREADY_GPS_THREAD_INITED : 이미 GPS 스레드가 초기화 되어 있다.

_HS_GPS_THREAD_START_FAULT : GPS 스레드 시작이 실패했다.

See Also

close_gps()

start_gps()

read_gps_data()

3.8 init_sensors

Synopsis

#include "hs_sensor_gps.h"

int init_sensors(*void*);

Description

센서를 사용하기 위해 초기화 하고, 센서 값을 주기적으로 읽는 Thread가 생성되지만 센서 값을 파일로 저장 하지 않는다. 또한, 이 상태에서 함수에 timestamp를 0으로 하여 현재 값을 읽을 수 있다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_PARAMETER_ERROR : 소스레벨 파라미터가 오류를 발생했다.

_HS_ALREADY_INITED : 이미 초기화 되어 있다.

_HS_THREAD_START_FAULT : Thread 시작 오류가 발생했다.

See Also

close_sensor()

start_sensor()

read_sensor_data()

3.9 read_gps_data

Synopsis

#include "hs_sensor_gps.h"

int read_gps_data(*time_ttimeStamp,struct* buf, int count);

Description

해당 시간의 데이터로부터 버퍼에 count 개수만큼의 데이터를 배열로 읽어온다. 만약 시간이 0으로 지정되면, 가장 최근 데이터를 하나 읽어오게 된다. 그리고 저장이 안 된 부분의 경우 0으로 채워진 데이터를 리턴 받게 된다.

Parameter

time_t timestamp : UTC epoch time 값.

struct * buf : _GPS_DATA 배열. (이 메모리는 호출하는 쪽에서 할당을 해야 한다.)

int count : 읽고자 하는 _GPS_DATA 개수.

Return Value

성공하면 실제 복사된 구조체의 개수를 반환한다. 대부분의 경우, 호출할 때의 count와 같을 것이며, 최근 데이터를 읽을 때만 실제 저장된 값이 없으므로 count보다 작은 값이 리턴 될 것이다. 실패하면 음수를 리턴 한다

Errors

_HS_INVALID_NUMBER : count가 양수가 아니다.

See Also

_GPS_DATA, delete_gps_data()

Example

```
#include "hs_sensor_gps.h"
int main(void)
{
  int count = 1;
  int error_number;

  struct * gps_data = (struct  *) malloc (sizeof(struct ) * count);
```

```
    error_number =  (0, gps_data, count) // 현재 값 읽기
    if (error_number < 0)
      // 에러 메시지 출력
    free(gps_data);
}
```

3.10 read_sensor_data

Synopsis

#include "hs_sensor_gps.h"

int read_sensor_data(*time_ttimeStamp,struct* buf, int count);

Description

해당 시간의 데이터로부터 버퍼에 count 개수만큼의 데이터를 배열로 읽어온다. 만약 시간이 0으로 지정되면, 가장 최근 데이터를 하나 읽어오게 된다. 그리고 저장이 안 된 부분의 경우 0으로 채워진 데이터를 리턴 받게 된다.

Parameter

time_t timestamp : UTC epoch time 값.

struct * buf : _SENSOR_DATA 배열. (이 메모리는 호출하는 쪽에서 할당을 해야 한다.)

int count : 읽고자 하는 _SENSOR_DATA 개수.

Return Value

성공하면 실제 복사된 구조체의 개수를 반환한다. 대부분의 경우, 호출할 때의 count와 같을 것이며, 최근 데이터를 읽을 때만 실제 저장된 값이 없으므로 count보다 작은 값이 리턴 될 것이다. 실패하면 음수를 리턴 한다.

Errors

_HS_INVALID_NUMBER : count가 양수가 아니다.

See Also

_SENSOR_DATA, delete_sensor_data() ,

Example

```
#include "hs_sensor_gps.h"
int main(void)
{
  int count = 1;
  int error_number;

  struct * sensor_data = (struct *) malloc (sizeof(struct ) * count);
```

```
    error_number =  (0, sensor_data, count) // 현재 값 읽기
    if (error_number < 0)
      // 에러 메시지 출력
    free(sensor_data);
}
```

3.11 register_gps_callback(void)

Synopsis

#include "hs_sensor_gps.h"

int **register_gps_callback**(*void (*func)(int pos_index), struct * pos, int count*);

Description

callback 함수와 좌표 배열을 등록하면 나중에 차가 그 좌표 중 어딘가에 도착하면 호출된다. 만약 이 함수를 등록할 때 이전 내용이 있으면 덮어쓴다.

Parameter

void (*func)(int pos_index) : 호출될 함수 포인터. (인자는 도착한 좌표의 함수에 당시의 index 값.)

struct * pos : callback 함수가 호출 되야 하는 지점들의 배열로 이 배열의 index를 인자로 등록된 callback 함수가 호출된다. (이 메모리는 호출하는 쪽에서 할당을 해야 한다.)

int count : struct 의 개수.

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_INVALID_FUNCTION_POINTER : callback 함수의 주소가 NULL이다.

See Also

cancel_gps_callback()

register_gps_callback()

Example

```
#include "hs_sensor_gps.h"
void  gps_callback_func (int pos_index)
{
  // gps 콜백 함수 실행
}
int main(void)
```

```
{
   int error_number;
   struct * gps_position = (struct *) malloc (sizeof(struct ));
   gps_position->latitude = 3735.0079;
   gps_position->longitude = 12701.6446;
   void (*func)(int) = gps_callback_func;
   error_number = (gps_callback_func, gps_position, 1); // 특정 좌표 리스트
에 대한 callback 함수를 지정한다.
   if (error_number < 0)
     // error 메시지 출력
   free(gps_position);
}
```

3.12 register_sensor_callback

Synopsis

#include "hs_sensor_gps.h"

int register_callback(*void(*func)(void*),struct*base*);

Description

Callback 함수와 를 등록하면 나중에base를 기준으로 더 큰 센서 값이 측정되면 호출된다. 만약 이 함수를 등록할 때 이전 내용이 있으면 덮어쓴다. 만약 의한 값이라도 0이면 그 vector는 callback 대상이 아니다.

Parameter

void (*func)(void*) : 호출될 함수 포인터.

struct * base : 기준이 될 센서 값. (이 메모리는 호출하는 쪽에서 할당을 해야 한다.)

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_INVALID_FUNCTION_POINTER : callback 함수의 주소가 NULL이다.

See Also

cancel_sensor_callback()

Example

```
#include "hs_sensor_gps.h"
void  sensor_callback_func (void)
{
  // sensor 콜백 함수 실행
}
int main(void)
{
  struct * acc_vector = (struct *) malloc (sizeof(struct ));
  acc_vector->x = 2;
  acc_vector->y = 2;
```

```
    acc_vector->z = 2;
    void (*func)( void) = sensor_callback_func;
    error_number = (sensor_callback_func, acc_vector); // 특정 센서 값에 대한
callback 함수를 지정한다.
    if (error_number < 0)
      // error 메시지 출력
    free(acc_vector);
 }
```

3.13 start_gps

Synopsis

#include “hs_sensor_gps.h”

int start_gps(*void*)

Description

데이터의 저장을 시작한다. 저장 주기는 header file의 소스레벨에서 정해지며, 데이터 저장 시 기록되는 시간은 4Byte UTC epoch sec로 저장된다. 기록되는 파일은

C:/GPS_DATA/YYYY/MM/dd/HH.txt

로, 한 시간 단위로 저장된다.

만약, 함수가 12시 10분 00초에 호출되었고 저장 주기가 1000ms였다면, 12시 00분 00초부터 12시 9분 59초까지 0이 채워진다. 만약 12시 02분 00초까지 데이터가 저장된 적이 있었다면, 12시 02분 01초부터 12시 9분 59초까지 0이 채워진다.

그리고 데이터는 “인덱스, 시간, 위도, 경도, 속도 1, 속도 2, 속도 3”로 저장되며 데이터 형식은 구조로 저장된다. 또한, 이 함수 전에 함수가 호출되어야 한다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_NOT_INITIALIZED: 초기화 된 적이 없다.

See Also

_GPS_DATA, cancel_gps_callback(), close_gps(),

init_gps(), stop_gps()

3.14 start_sensor

Synopsis

#include “hs_sensor_gps.h”

int **start_sensor**(*void*)

Description

데이터의 저장을 시작한다. 저장 주기는 소스레벨에서 정해지며, 데이터 저장 시 기록되는 시간은 4Byte UTC epoch sec로 저장된다. 기록되는 파일은

C:/SENSOR_DATA/YYYY/MM/dd/HH.txt

로, 한 시간 단위로 저장된다.

만약,) 함수가 12시 10분 00초에 호출되었고 저장 주기가 1000ms였다면, 12시 00분 00초부터 12시 9분 59초까지 0이 채워진다. 만약 12시 02분 00초까지 데이터가 저장된 적이 있었다면, 12시 02분 01초부터 12시 9분 59초까지 0이 채워진다.

그리고 데이터는 “시간, 운전 점수, 가속도 센서 x축의 벡터 값, 가속도 센서 y축의 벡터 값, 가속도 센서 z축의 벡터 값, 최근 10초 동안의 가속도 센서 x축의 벡터 값, 최근 10초 동안의 가속도 센서 y축의 벡터 값, 최근 10초 동안의 가속도 센서 z축의 벡터 값, 최근 1분 동안의 가속도 센서 x축의 벡터 값, 최근 1분 동안의 가속도 센서 y축의 벡터 값, 최근 1분 동안의 가속도 센서 z축의 벡터 값”로 저장되며 데이터 형식은 구조로 저장된다. 또한, 이 함수 전에 함수가 호출되어야 한다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_FILE_SAVE_FAULT : 파일 저장을 실패했다.

_HS_PARAMETER_ERROR : 소스레벨 파라미터가 오류를 발생했다.

_HS_NOT_INITIALIZED: 초기화 된 적이 없다.

See Also

_SENSOR_DATA

cancel_sensor_callback()

close_sensor()

init_sensors()

stop_sensor

3.15 stat_gps_data

Synopsis

#include "hs_sensor_gps.h"

int stat_gps_data(*struct* gps_data_info*);

Description

gps 데이터 정보를 반환한다. 이를 통하여 저장주기(msec)와, 파일로 저장된 정보의 최초 시간, 가장 최근 저장시간 등을 확인할 수 있다.

Parameter

struct * gps_data_info : 저장주기, 최초시간, 최근 저장시간이 포함된 gps 데이터 정보이면 만약 포함된 값들이 0인 경우에는 함수가 아직 실행되지 않아 초기값이 리턴한다. (이 메모리는 호출하는 쪽에서 할당을 해야 한다.)

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_NO_DATAFILE : 데이터 파일이 존재하지 않습니다.

See Also

_GPS_DATA_INFO

Example

```
#include "hs_sensor_gps.h"
int main(void)
{
    int error_number;
      struct * gps_data_info = (struct *) malloc (sizeof(struct ));

  error_number = (gps_data_info); // gps 데이터 정보를 반환한다
  if (error_number < 0)
    // error 메시지 출력
  printf("저장주기 : %d, 최초 저장시간 %d, 최근 저장시간 %d",
gps_data_info -> saveInterval, gps_data_info -> startTimeStamp,
```

```
gps_data_info -> endTimeStamp);
    free(gps_data_info);
 }
```

3.16 stat_sensor_data

Synopsis

#include "hs_sensor_gps.h"

int stat_sensor_data(struct* sensor_data_info)

Description

센서 데이터 정보를 반환합니다. 이를 통하여 저장주기(msec)와, 파일로 저장된 정보의 최초 시간, 가장 최근 저장시간 등을 확인할 수 있다.

Parameter

struct * sensor_data_info : 저장주기, 최초시간, 최근 저장시간이 포함된 sensor 데이터 정보이면 만약 포함된 값들이 0인 경우에는 함수가 아직 실행되지 않아 초기값이 리턴. (이 메모리는 호출하는 쪽에서 할당을 해야 한다.)

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_NO_DATAFILE : 데이터 파일이 존재하지 않습니다.

See Also

_SENSOR_DATA_INFO

Example

```
#include "hs_sensor_gps.h"
int main(void)
{
    int error_number;
  struct * sensor_data_info = (struct *) malloc (sizeof(struct ));

  error_number = (sensor_data_info); // sensor 데이터 정보를 반환한다
  if (error_number < 0)
    // error 메시지 출력
  printf("저장주기 : %d, 최초 저장시간 %d, 최근 저장시간 %d",
sensor_data_info -> saveInterval, sensor_data_info -> startTimeStamp,
```

```
sensor_data_info -> endTimeStamp);
    free(sensor_data_info);
}
```

3.17 stop_gps

Synopsis

#include “hs_sensor_gps.h”

int stop_gps(*void*)

Description

gps 데이터 저장을 중단하고, gps 파일을 기록 후 파일을 닫습니다. 또한, GPS 데이터를 읽어오는 스레드를 종료하기 위해서는 함수를 호출해야 합니다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_FILE_SAVE_FAULT : 파일 저장을 실패했다.

_HS_NOT_INITIALIZED: 초기화 된 적이 없다.

See Also

close_gps()

start_gps()

3.18 stop_sensor

Synopsis

#include "hs_sensor_gps.h"

int stop_sensor(*void*)

Description

센서 데이터 저장을 중지하고, 데이터 파일을 기록 후 파일을 닫습니다. 또한, 센서를 읽어오는 스레드를 종료하기 위해서는 함수를 호출해야 합니다.

Parameter

NONE

Return Value

성공하면 0 (_HS_TRUE), 실패하면 음수인 에러코드를 return한다.

Errors

_HS_FILE_SAVE_FAULT : 파일 저장을 실패했다.

_HS_NOT_INITIALIZED: 초기화 된 적이 없다.

See Also

close_sensor()

start_sensor()

ABSTRACT

Driving Stability Management System Using acceleration sensor and GPS.

Kim, Sunho
Major in Computer Engineering
Dept. of Computer Engineering
Graduate School, Hansung University

According to the accident statistics of 2009, it can be recognized that drivers characteristics and driving behaviors are the most causational factors on the traffic accidents. also the number of commercial vehicle(city, suburb and other buses) accidents consumes 18.4 percent of the total number of traffic accidents in this year. Since the commercial vehicles are responsible for not only the drivers but also the passengers, it leads more serious social and economic problems. There have been various forms of systems such as a digital speedometer or a black box to meet the social requirement for reducing traffic accidents and safe driving. however the system based on the data after accident control the driver by analyze dangerous drive behaviors, so there is a limit to provide the driver warning information in real-time. In this study performed Driving Stability Management System for drivers using acceleration sensor and GPS.